ar_submit

NAME

qrsub -

Submit an Advance Reservation (AR) to Altair Grid Engine.

qralter -

Modify an Advance Reservation (AR) of Altair Grid Engine.

SYNTAX

qrsub options

qralter options ar_id|ar_name

DESCRIPTION

Qrsub provides a means for operators, managers, or users listed in the ACL (see *sge_access_list*(5)) "**arusers**" to create an Advance Reservation (AR) or a Standing Reservation (SR) in the Altair Grid Engine queuing system. ARs allow reservation of particular consumable resources for future use. These reserved resources are only available to jobs requesting the AR, and the scheduler ensures the availability of the resources when the start time is reached. Jobs requesting the AR can only use the reserved consumable resources. SRs are recurring ARs which follow a given calendar. All ARs within one SR have the same ID.

During AR submit time the Altair Grid Engine queuing system selects the best-suited queues for the AR request, and then reserves the desired amount of resources. For a reservation, all queues that are not in an orphaned state are considered to be suited. The AR will be granted only if the AR request can be fulfilled.

ARs will be deleted either automatically when the end time is reached, or manually using *qrdel*. In both cases, first all jobs requesting the AR will be removed, and then the AR itself. Already-granted ARs can be shown with *qrstat*.

Note: To make AR behavior predictable, it is necessary to have reserved resources available at the time of AR start. This is done by keeping jobs with an unlimited runtime limit separated from ARs, and not considering resources used by such jobs for reservation.

Note: Resource Quotas are not considered for AR queue selection or for jobs requesting an AR.

When an AR is successfully added to the Altair Grid Engine queuing system, *qrsub* returns a unique integer ID referring the the newly-created AR. The highest AR ID is 9999999. If the

highest ID is reached, a wraparound happens and the next unused ID, starting with 1, will be used.

For *qrsub*, the administrator and the user may define default request files (analogous to Altair Grid Engine_request for qsub), which can contain any of the possible command-line options.
A cluster-wide default request file is optional. If such a default request file is used, it must be placed under
$SGE_ROOT/$SGE_CELL/common/sge_ar_request (global defaults file).
A user-private default request file is optional. If it is used, it must be placed under
$HOME/.sge_ar_request (user private defaults file).

qralter can be used to change the attributes of an advance reservation. Whether an attribute can be changed depends on the state of the AR:

All attributes can be changed for ARs that are pending or running, but there must be no jobs running in the AR. It might become necessary to reschedule the AR, e.g. if resource requests (-l / -masterl) or queue requests (-q / -masterq) are modified. If rescheduling is not possible because the requested resources are not available in the given time frame, qralter will print an error message and the AR will not be modified.

If an AR is already running and has running jobs:

- simple modifications not affecting the reserved resources, such as modifying the name (-N) or the account string (-A), will always work
- modifying start time (-a), end time (-e), or duration (-d) will work, if the resources held by the AR will also be available in the new time frame. Reducing the time frame will always be accepted.
- if rescheduling of the AR would be necessary because e.g. resource requests will be modified (-l / -masterl) or the given set of resources will not be available in an extended time frame (-e / -d), qralter will print an error message and the AR will not be modified.

With the qmaster_param AR_DETACH_JOBS_ON_RESCHEDULE it is possible to have running jobs being detached from the AR and thus allowing re-scheduling; see sge_conf(5).

OPTIONS

-a date_time

Defines the activation (start) date and time of an AR. The option is not required. If it is omitted, the current date_time is assumed. Either a duration or end date_time must be specified for ARs. For details about date_time please see *sge_types*(1) For SRs the start time is optional and determines the earliest possible allocation time of an AR (but not the actual beginning of the first AR).

-A account_string

Identifies the account to which the resource reservation of the AR should be charged. For "**account_string**" value details please see the "**name**" definition in *sge_types*(1). In the ab-

sence of this parameter Altair Grid Engine will place the default account string “sge” in the accounting record of the AR.

-cal_week weekly_calendar

If a calendar is defined it is a Standing Reservation request. The weekly calendar has the same format as the calendar object (see *sge_calendar_conf*(5)) with the exception that only the “**on**” state is allowed. The weekly calendar defines the time ranges for which AR instances should be created. The scheduler tries to allocate a fixed amount of ARs. That amount can be influenced with the depth (“**-cal_depth**”) parameter. All ARs within an SR have the same ID. Jobs running within one AR are deleted at AR end (as with ARs) but queued jobs stay queued waiting for the next AR occurrence within the SR. Giving a start time, end time, or duration is optional for SRs. If given, the start time determines the earliest allowed start time for the first AR. The end time determines the last possible end time of the last AR within the SR. The duration specifies the end time. If one AR within the SR ends, new ARs are scheduled automatically, so that the total amount of allocated ARs are equal to the SR depth.

-cal_depth number

Specifies the depth of the SR. The depth determines how many ARs are allocated (scheduled) at any point in time in the future. ARs which cannot get enough resources are unallocated and not counted. If one AR ends, further ARs are allocated so that the depth is restored. If an AR cannot be allocated it will go into an Error state. Default for “**-cal_depth**” is 8.

-cal_jmp number

The jmp parameter determines how many of the first calendar entries can be jumped over if they cannot be allocated for ARs due to missing resources at SR submission time. If the first n ARs of a standing reservation cannot be allocated and “**-cal_jmp**” is less than n, the submission of the SR will be rejected. Default for “**-cal_jmp**” is 0.

-ckpt ckpt_name

Selects the checkpointing environment (see *sge_checkpoint*(5)) the AR jobs may request. Using this option guarantees that only queues providing this checkpoint environment will be reserved.

-d time

Defines the duration of the AR. The use of “**-d time**” is optional if “**-e date_time**” is requested. For details about “**time**” definition please see *sge_types*(1). For SRs the duration is optional and determines implicitly the end time, i.e. the last possible end time for the last AR within the SR.

-e date_time

Defines the end date and time of an AR. The use of "**-e date_time**" is optional if "**-d time**" is requested. For details about "**date_time**" definition please see *sge_types*(1). For SRs the end time is optional and determines the last possible end time for the last AR within the SR.

-fr y[es]|n[o]

Specifies the behavior of the AR at AR start time. With ("**-fr yes**") jobs still blocking resources being granted to the AR will be killed. With default behavior ("**-fr no**") jobs holding resources being granted to the AR can continue running and may prevent start of jobs in the AR. In the current implementation only resources requested by the AR via exclusive complexes are freed. The feature is suited for freeing whole queue instances or hosts or even the whole cluster, e.g. for maintenance purposes. Requesting "**-fr yes**" requires operator privileges.

-he y[es]|n[o]

Specifies the behavior when the AR goes into an error state. The AR goes into an error state when a reserved host goes into an unknown state, a queue error happens, or when a queue is disabled or suspended.

A hard error, "**-he yes**", means as long as the AR is in an error state no jobs using the AR will be scheduled. If a soft error, "**-he no**", is specified the AR remains usable with the remaining resources.
By default soft error handling is used.

-help

Prints a list of all options.

-version

Display version information for the command

-jc IGNORE_JC | NO_JC | ANY_JC |

If specified allows filtering of queue instances that may be considered for AR reservation.
By default, if the **-jc** switch is omitted or when **-jc IGNORE_JC** is specified, queues that accept JC **and** non-JC jobs at the same time will be considered for the AR. If the only queues that should be selected are those that do not allow execution of JC jobs, **-jc NO_JC** can be specified.
If the only queues that should be selected are those that allow execution of JC jobs, **-jc ANY_JC** is the right choice.
Additionally it is possible to filter queue instances that accept jobs derived from specific job classes. To achieve this a pattern for job class variants, or a specific job class variant name,

can be specified. This will select only those queue instances where jobs that are derived from a job class variant that matches the pattern are also allowed to be executed.

-l resource=value,...

Creates an AR in a Altair Grid Engine queue; the AR provides the given resource request list. *sge_complex*(5) describes how a list of available resources and their associated valid value specifiers can be obtained.
There may be multiple **-l** switches in a single command.

With AR submission it is possible to overwrite the consumable attribute of a resource description (the complex variable). Overwriting a consumable is only possible for complex variables which are already consumable (one of YES, JOB, HOST).

Syntax: -l resource[=value[{consumable}]][,resource[=value[{consumable}]],...] where consumable is one of YES, JOB, HOST.

Can be combined with **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

-masterl resource=value,...

Available only in combination with parallel jobs, i.e. together with the -pe option.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

qralter does allow changing the value of this option for the reservation; however the modification will only be effective if the job is not running in the reservation.

-masterq wc_queue_list

Available only in combination with parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may

make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qralter does allow changing the value of this option for the reservation; however the modification will only be effective if the job is not running in the reservation.

-m b|e|a|n

Defines or redefines under which circumstances mail is to be sent to the AR owner or to the users listed with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the AR
'e' Mail is sent at the end of the AR
'a' Mail is sent when the AR goes into an error state
'n' No mail is sent. This is the default for *qrsub*

-M user[@host],...

Defines or redefines the list of users to which the qmaster sends mail.

-masterq wc_queue_list

Only meaningful for a parallel AR request when used together with the -pe option.

This option is used to reserve the proper queues to match this request as if it were requested via qsub. A more detailed description of *wc_queue_list* can be found in *sge_types*(1).

-now y[es]|n[o]

This options impacts the queues selection for reservation.
With the "**-now y**" option, only queues with the qtype "INTERACTIVE" assigned will be considered for reservation. "**-now n**" is the default for *qrsub*.

-N name

The name of the AR. The name, if requested, must conform to "**name**" as defined in *sge_types*(1). Invalid names will be denied at submit time.

Note that names have to be unique when the qmaster param **SUBMIT_JOBS_WITH_AR_NAME** is set to true. qralter cannot change the AR name when using ar_name as an argument.

-P project_name

This option binds the AR to a project. Only members of this project are allowed to use this AR.

-w e|v

Specifies the validation level applied to the AR request.

The specifiers e and v define the following validation modes:

'v' Verify: does not submit the AR but prints an extensive validation report

'e' Error: rejects request if requirements cannot be fulfilled. This is the default for *qrsub*

-par allocation_rule

This option can be used with a parallel programming environment (PE).

It can be used to overwrite the allocation rule of the parallel environment into which an AR gets submitted with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel advance reservation.

This option can also be used after a **-petask** switch to overwrite the allocation rule for a specific set of tasks only. If there are multiple task sets defined at the command line, the scheduler will combine all groups with identical allocation rules. The tasks of such a group will be scheduled together. The scheduler will start with the group containing the task with the smallest ID, then will continue with the next group containing the smallest unscheduled task.

Valid allocation rules are **$pe_slots**, **$fill_up**, **$round_robin**, and a positive number as a **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

-pe parallel_env n[-[m]]|[-]m,...

Parallel programming environment (PE) to select for the AR queue reservation. Please see the details of a PE in *sge_pe*(5).

-petask tid_range_list ...

Available for *qrsub* and *qralter*.

Can be used to submit parallel advance reservations (submitted with **-pe** request); indicates that all resource requirements specified via **-q** and **-l**, and in combination with **-hard**, **-soft**, and **-par** that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the master task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request, the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The master task (ID 0) has a memory request of 4M. All slave tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

qralter can be used in combination with the **-petask** switch to completely replace the previously specified requests for an existing task ID range, as long as no additional restrictions

apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page. Please note also that attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission will be rejected.

-q wc_queue_list

Defines or redefines a list of cluster queues, queue domains, or queue instances that may be reserved by the AR. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-petask** to specify specific queue requirements for individual PE tasks or a group of PE tasks. Find more information in the corresponding sections of this man page.

-rao y[es]|n[o]

Specifies which resources will be considered for the AR at submission/reservation time. When using **-rao yes**, only currently available resources will be reserved. This means using only queues which are not in disabled, suspended, error or unknown state. The default setting for **-rao** is "no". A cluster-wide setting for only reserving available resources can be done via qmaster_param AR_RESERVE_AVAILABLE_ONLY; see *sge_conf*(5).

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-u [username|@access_list],...

Specifies the users allowed to submit jobs requesting the AR. The access is specified by a comma-separated list containing UNIX users or ACLs (see *sge_access_list*(5)). An ACL name is prefixed with an '@' sign.
By default only the AR owner is allowed to submit jobs requesting the AR.

Note: Only queues where all users specified in the list have access are considered for reservation (see *sge_queue_conf*(5)).

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell, *qsub*, *qsh*, *qlogin*, or *qalter* use (in order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the TCP port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead of defining the port.

FILES

$SGE_ROOT/$SGE_CELL/common/sge_ar_request

global defaults file

$HOME/.sge_ar_request

user-private defaults file

SEE ALSO

qrdel(1), *qrstat*(1), *qsub*(1), *sge_types*(1), *sge_checkpoint*(5), *sge_complex*(5), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_pe*(5), *sge_resource_quota*(5), *sge_calendar_conf*(5).

COPYRIGHT

gethostbyaddr

NAME

gethostname - get local hostname.

gethostbyname - get local host information for specified hostname.

gethostbyaddr - get hostname via IP address.

getservbyname - get configured port number of service

SYNTAX

gethostname [-help|-name|-aname|-all]

gethostbyname [-help|-name|-aname|-all|-all_rr|-verify_alias_file] <name|alias_file_path>

gethostbyaddr [-help|-name|-aname|-all]

getservbyname [-help|-number]

DESCRIPTION

gethostname and *gethostbyname* are used to get the local resolved host name. *gethostbyaddr* is used to get the hostname of a specified IP address (dotted decimal notation). *getservbyname* can be used to get the configured port number of a service (e.g. from /etc/services). The *gethostbyname* output for **-all** and **-all_rr** will also show the used method for resolving the hostname. The method is set to "OS" if the host was resolved via system call. Another value can be "H2IP4" which is set if the host was resolved by parsing the hostname for the IPv4 address.

The *gethostbyname* option **-verify_alias_file** can be used to parse the specified path to a host_aliases file (See also *sge_host_aliases*(5) man page). If the file could be parsed without errors the exit value of *gethostbyname* will be 0. Detected errors in the host_aliases file will be printed to stderr and the exit value will be set to 2. The verification will also check if the resulting alias name is resolvable on the host where *gethostbyname* was started.

The hostname utils are primarily used by the Altair Grid Engine installation scripts. *gethostname*, *gethostbyname* and *gethostbyaddr* called without any option will print out the hostname, all specified aliases, and the IP address of the locally resolved hostname. Calling *getservbyname* without any option will print out the full service entry.

OPTIONS

-help

Prints a list of all options.

-version

Display version information for the command.

-name

This option only reports the primary name of the host.

-aname

If this option is set, the Altair Grid Engine host alias file is used for host name resolving. It is necessary to set the environment variable SGE_ROOT and, if more than one cell is defined, also SGE_CELL.

This option will print out the Altair Grid Engine host name.

-all

By using the **-all** option all available host information will be printed. This information includes the host name, the Altair Grid Engine host name, all host aliases, and the IP address of the host.

-all_rr (only for gethostbyname)

The **-all_rr** option will enhance the **-all** option output. It will in addition try to contact the sge_qmaster daemon in order to resolve the specified hostname also on the sge_qmaster host.

If the daemon is not running or cannot be contacted the command will print out errors and terminate with an error exit code.

If the sge_qmaster host could be contacted successfully the output will include an extra line showing the resulting hostname on the sge_qmaster host and also the host name of the sge_qmaster service.

-number

This option will print out the port number of the specified service name.

<name>

The host name for which the information is requested.

<ip>

The IP address (dotted decimal notation) for which the information is requested.

<service>

The service name for which the information is requested (e.g. ftp, sge_qmaster or sge_execd).

EXAMPLES

The following example shows how to get the port number of the FTP service:

```
>getservbyname -number ftp
21
```

The next example shows the output of gethostname -all when the host alias file contains this line:

gridmaster extern_name extern_name.mydomain

The local host resolving must also provide the alias name “gridmaster”. Each Altair Grid Engine host that wants to use the cluster must be able to resolve the host name “gridmaster”.

To setup an alias name, edit your /etc/hosts file or modify your NIS setup to provide the alias for the NIS clients.

The host alias file must be readable from each host (use e.g. NFS shared file location).

```
>gethostname -all
Hostname: extern_name.mydomain
SGE name: gridmaster
Aliases:  loghost gridmaster
Host Address(es): 192.168.143.99
```

ENVIRONMENTAL VARIABLES

SGE_ROOT Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL If set, specifies the default Altair Grid Engine cell.

SEE ALSO

sge_intro(1), *sge_host_aliases*(5)

COPYRIGHT

gethostbyname

NAME

gethostname - get local hostname.

gethostbyname - get local host information for specified hostname.

gethostbyaddr - get hostname via IP address.

getservbyname - get configured port number of service

SYNTAX

gethostname [-help|-name|-aname|-all]

gethostbyname [-help|-name|-aname|-all|-all_rr|-verify_alias_file] <name|alias_file_path>

gethostbyaddr [-help|-name|-aname|-all]

getservbyname [-help|-number]

DESCRIPTION

gethostname and *gethostbyname* are used to get the local resolved host name. *gethostbyaddr* is used to get the hostname of a specified IP address (dotted decimal notation). *getservbyname* can be used to get the configured port number of a service (e.g. from /etc/services). The *gethostbyname* output for **-all** and **-all_rr** will also show the used method for resolving the hostname. The method is set to "OS" if the host was resolved via system call. Another value can be "H2IP4" which is set if the host was resolved by parsing the hostname for the IPv4 address.

The *gethostbyname* option **-verify_alias_file** can be used to parse the specified path to a host_aliases file (See also *sge_host_aliases*(5) man page). If the file could be parsed without errors the exit value of *gethostbyname* will be 0. Detected errors in the host_aliases file will be printed to stderr and the exit value will be set to 2. The verification will also check if the resulting alias name is resolvable on the host where *gethostbyname* was started.

The hostname utils are primarily used by the Altair Grid Engine installation scripts. *gethostname*, *gethostbyname* and *gethostbyaddr* called without any option will print out the hostname, all specified aliases, and the IP address of the locally resolved hostname. Calling *getservbyname* without any option will print out the full service entry.

OPTIONS

-help

Prints a list of all options.

-version

Display version information for the command.

-name

This option only reports the primary name of the host.

-aname

If this option is set, the Altair Grid Engine host alias file is used for host name resolving. It is necessary to set the environment variable SGE_ROOT and, if more than one cell is defined, also SGE_CELL.

This option will print out the Altair Grid Engine host name.

-all

By using the **-all** option all available host information will be printed. This information includes the host name, the Altair Grid Engine host name, all host aliases, and the IP address of the host.

-all_rr (only for gethostbyname)

The **-all_rr** option will enhance the **-all** option output. It will in addition try to contact the sge_qmaster daemon in order to resolve the specified hostname also on the sge_qmaster host.

If the daemon is not running or cannot be contacted the command will print out errors and terminate with an error exit code.

If the sge_qmaster host could be contacted successfully the output will include an extra line showing the resulting hostname on the sge_qmaster host and also the host name of the sge_qmaster service.

-number

This option will print out the port number of the specified service name.

<name>

The host name for which the information is requested.

<ip>

The IP address (dotted decimal notation) for which the information is requested.

<service>

The service name for which the information is requested (e.g. ftp, sge_qmaster or sge_execd).

EXAMPLES

The following example shows how to get the port number of the FTP service:

```
>getservbyname -number ftp
21
```

The next example shows the output of gethostname -all when the host alias file contains this line:

gridmaster extern_name extern_name.mydomain

The local host resolving must also provide the alias name “gridmaster”. Each Altair Grid Engine host that wants to use the cluster must be able to resolve the host name “gridmaster”.

To setup an alias name, edit your /etc/hosts file or modify your NIS setup to provide the alias for the NIS clients.

The host alias file must be readable from each host (use e.g. NFS shared file location).

```
>gethostname -all
Hostname: extern_name.mydomain
SGE name: gridmaster
Aliases:  loghost gridmaster
Host Address(es): 192.168.143.99
```

ENVIRONMENTAL VARIABLES

SGE_ROOT Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL If set, specifies the default Altair Grid Engine cell.

SEE ALSO

sge_intro(1), *sge_host_aliases*(5)

COPYRIGHT

gethostname

NAME

gethostname - get local hostname.

gethostbyname - get local host information for specified hostname.

gethostbyaddr - get hostname via IP address.

getservbyname - get configured port number of service

SYNTAX

gethostname [-help|-name|-aname|-all]

gethostbyname [-help|-name|-aname|-all|-all_rr|-verify_alias_file] <name|alias_file_path>

gethostbyaddr [-help|-name|-aname|-all]

getservbyname [-help|-number]

DESCRIPTION

gethostname and *gethostbyname* are used to get the local resolved host name. *gethostbyaddr* is used to get the hostname of a specified IP address (dotted decimal notation). *getservbyname* can be used to get the configured port number of a service (e.g. from /etc/services). The *gethostbyname* output for **-all** and **-all_rr** will also show the used method for resolving the hostname. The method is set to "OS" if the host was resolved via system call. Another value can be "H2IP4" which is set if the host was resolved by parsing the hostname for the IPv4 address.

The *gethostbyname* option **-verify_alias_file** can be used to parse the specified path to a host_aliases file (See also *sge_host_aliases*(5) man page). If the file could be parsed without errors the exit value of *gethostbyname* will be 0. Detected errors in the host_aliases file will be printed to stderr and the exit value will be set to 2. The verification will also check if the resulting alias name is resolvable on the host where *gethostbyname* was started.

The hostname utils are primarily used by the Altair Grid Engine installation scripts. *gethostname*, *gethostbyname* and *gethostbyaddr* called without any option will print out the hostname, all specified aliases, and the IP address of the locally resolved hostname. Calling *getservbyname* without any option will print out the full service entry.

OPTIONS

-help

Prints a list of all options.

-version

Display version information for the command.

-name

This option only reports the primary name of the host.

-aname

If this option is set, the Altair Grid Engine host alias file is used for host name resolving. It is necessary to set the environment variable SGE_ROOT and, if more than one cell is defined, also SGE_CELL.

This option will print out the Altair Grid Engine host name.

-all

By using the **-all** option all available host information will be printed. This information includes the host name, the Altair Grid Engine host name, all host aliases, and the IP address of the host.

-all_rr (only for gethostbyname)

The **-all_rr** option will enhance the **-all** option output. It will in addition try to contact the sge_qmaster daemon in order to resolve the specified hostname also on the sge_qmaster host.

If the daemon is not running or cannot be contacted the command will print out errors and terminate with an error exit code.

If the sge_qmaster host could be contacted successfully the output will include an extra line showing the resulting hostname on the sge_qmaster host and also the host name of the sge_qmaster service.

-number

This option will print out the port number of the specified service name.

<name>

The host name for which the information is requested.

<ip>

The IP address (dotted decimal notation) for which the information is requested.

<service>

The service name for which the information is requested (e.g. ftp, sge_qmaster or sge_execd).

EXAMPLES

The following example shows how to get the port number of the FTP service:

```
>getservbyname -number ftp
21
```

The next example shows the output of gethostname -all when the host alias file contains this line:

gridmaster extern_name extern_name.mydomain

The local host resolving must also provide the alias name “gridmaster”. Each Altair Grid Engine host that wants to use the cluster must be able to resolve the host name “gridmaster”.

To setup an alias name, edit your /etc/hosts file or modify your NIS setup to provide the alias for the NIS clients.

The host alias file must be readable from each host (use e.g. NFS shared file location).

```
>gethostname -all
Hostname: extern_name.mydomain
SGE name: gridmaster
Aliases:  loghost gridmaster
Host Address(es): 192.168.143.99
```

ENVIRONMENTAL VARIABLES

SGE_ROOT Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL If set, specifies the default Altair Grid Engine cell.

SEE ALSO

sge_intro(1), *sge_host_aliases*(5)

COPYRIGHT

getservbyname

NAME

gethostname - get local hostname.

gethostbyname - get local host information for specified hostname.

gethostbyaddr - get hostname via IP address.

getservbyname - get configured port number of service

SYNTAX

gethostname [-help|-name|-aname|-all]

gethostbyname [-help|-name|-aname|-all|-all_rr|-verify_alias_file] <name|alias_file_path>

gethostbyaddr [-help|-name|-aname|-all]

getservbyname [-help|-number]

DESCRIPTION

gethostname and *gethostbyname* are used to get the local resolved host name. *gethostbyaddr* is used to get the hostname of a specified IP address (dotted decimal notation). *getservbyname* can be used to get the configured port number of a service (e.g. from /etc/services). The *gethostbyname* output for **-all** and **-all_rr** will also show the used method for resolving the hostname. The method is set to "OS" if the host was resolved via system call. Another value can be "H2IP4" which is set if the host was resolved by parsing the hostname for the IPv4 address.

The *gethostbyname* option **-verify_alias_file** can be used to parse the specified path to a host_aliases file (See also *sge_host_aliases*(5) man page). If the file could be parsed without errors the exit value of *gethostbyname* will be 0. Detected errors in the host_aliases file will be printed to stderr and the exit value will be set to 2. The verification will also check if the resulting alias name is resolvable on the host where *gethostbyname* was started.

The hostname utils are primarily used by the Altair Grid Engine installation scripts. *gethostname*, *gethostbyname* and *gethostbyaddr* called without any option will print out the hostname, all specified aliases, and the IP address of the locally resolved hostname. Calling *getservbyname* without any option will print out the full service entry.

OPTIONS

-help

Prints a list of all options.

-version

Display version information for the command.

-name

This option only reports the primary name of the host.

-aname

If this option is set, the Altair Grid Engine host alias file is used for host name resolving. It is necessary to set the environment variable SGE_ROOT and, if more than one cell is defined, also SGE_CELL.

This option will print out the Altair Grid Engine host name.

-all

By using the **-all** option all available host information will be printed. This information includes the host name, the Altair Grid Engine host name, all host aliases, and the IP address of the host.

-all_rr (only for gethostbyname)

The **-all_rr** option will enhance the **-all** option output. It will in addition try to contact the sge_qmaster daemon in order to resolve the specified hostname also on the sge_qmaster host.

If the daemon is not running or cannot be contacted the command will print out errors and terminate with an error exit code.

If the sge_qmaster host could be contacted successfully the output will include an extra line showing the resulting hostname on the sge_qmaster host and also the host name of the sge_qmaster service.

-number

This option will print out the port number of the specified service name.

<name>

The host name for which the information is requested.

<ip>

The IP address (dotted decimal notation) for which the information is requested.

<service>

The service name for which the information is requested (e.g. ftp, sge_qmaster or sge_execd).

EXAMPLES

The following example shows how to get the port number of the FTP service:

```
>getservbyname -number ftp
21
```

The next example shows the output of gethostname -all when the host alias file contains this line:

gridmaster extern_name extern_name.mydomain

The local host resolving must also provide the alias name “gridmaster”. Each Altair Grid Engine host that wants to use the cluster must be able to resolve the host name “gridmaster”.

To setup an alias name, edit your /etc/hosts file or modify your NIS setup to provide the alias for the NIS clients.

The host alias file must be readable from each host (use e.g. NFS shared file location).

```
>gethostname -all
Hostname: extern_name.mydomain
SGE name: gridmaster
Aliases:  loghost gridmaster
Host Address(es): 192.168.143.99
```

ENVIRONMENTAL VARIABLES

SGE_ROOT Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL If set, specifies the default Altair Grid Engine cell.

SEE ALSO

sge_intro(1), *sge_host_aliases*(5)

COPYRIGHT

hostnameutils

NAME

gethostname - get local hostname.

gethostbyname - get local host information for specified hostname.

gethostbyaddr - get hostname via IP address.

getservbyname - get configured port number of service

SYNTAX

gethostname [-help|-name|-aname|-all]

gethostbyname [-help|-name|-aname|-all|-all_rr|-verify_alias_file] <name|alias_file_path>

gethostbyaddr [-help|-name|-aname|-all]

getservbyname [-help|-number]

DESCRIPTION

gethostname and *gethostbyname* are used to get the local resolved host name. *gethostbyaddr* is used to get the hostname of a specified IP address (dotted decimal notation). *getservbyname* can be used to get the configured port number of a service (e.g. from /etc/services). The *gethostbyname* output for **-all** and **-all_rr** will also show the used method for resolving the hostname. The method is set to "OS" if the host was resolved via system call. Another value can be "H2IP4" which is set if the host was resolved by parsing the hostname for the IPv4 address.

The *gethostbyname* option **-verify_alias_file** can be used to parse the specified path to a host_aliases file (See also *sge_host_aliases*(5) man page). If the file could be parsed without errors the exit value of *gethostbyname* will be 0. Detected errors in the host_aliases file will be printed to stderr and the exit value will be set to 2. The verification will also check if the resulting alias name is resolvable on the host where *gethostbyname* was started.

The hostname utils are primarily used by the Altair Grid Engine installation scripts. *gethostname*, *gethostbyname* and *gethostbyaddr* called without any option will print out the hostname, all specified aliases, and the IP address of the locally resolved hostname. Calling *getservbyname* without any option will print out the full service entry.

OPTIONS

-help

Prints a list of all options.

-version

Display version information for the command.

-name

This option only reports the primary name of the host.

-aname

If this option is set, the Altair Grid Engine host alias file is used for host name resolving. It is necessary to set the environment variable SGE_ROOT and, if more than one cell is defined, also SGE_CELL.

This option will print out the Altair Grid Engine host name.

-all

By using the **-all** option all available host information will be printed. This information includes the host name, the Altair Grid Engine host name, all host aliases, and the IP address of the host.

-all_rr (only for gethostbyname)

The **-all_rr** option will enhance the **-all** option output. It will in addition try to contact the sge_qmaster daemon in order to resolve the specified hostname also on the sge_qmaster host.

If the daemon is not running or cannot be contacted the command will print out errors and terminate with an error exit code.

If the sge_qmaster host could be contacted successfully the output will include an extra line showing the resulting hostname on the sge_qmaster host and also the host name of the sge_qmaster service.

-number

This option will print out the port number of the specified service name.

<name>

The host name for which the information is requested.

<ip>

The IP address (dotted decimal notation) for which the information is requested.

<service>

The service name for which the information is requested (e.g. ftp, sge_qmaster or sge_execd).

EXAMPLES

The following example shows how to get the port number of the FTP service:

```
>getservbyname -number ftp
21
```

The next example shows the output of gethostname -all when the host alias file contains this line:

gridmaster extern_name extern_name.mydomain

The local host resolving must also provide the alias name “gridmaster”. Each Altair Grid Engine host that wants to use the cluster must be able to resolve the host name “gridmaster”.

To setup an alias name, edit your /etc/hosts file or modify your NIS setup to provide the alias for the NIS clients.

The host alias file must be readable from each host (use e.g. NFS shared file location).

```
>gethostname -all
Hostname: extern_name.mydomain
SGE name: gridmaster
Aliases:  loghost gridmaster
Host Address(es): 192.168.143.99
```

ENVIRONMENTAL VARIABLES

SGE_ROOT Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL If set, specifies the default Altair Grid Engine cell.

SEE ALSO

sge_intro(1), *sge_host_aliases*(5)

COPYRIGHT

qacct

NAME

qacct - report and account for Altair Grid Engine usage

SYNOPSIS

qacct [**-ar [ar_id|ar_name]**] [**-A [account_string]**] [**-b begin_time**] [**-d days**] [**-e end_time**] [**-g [group_id|group_name]**] [**-h [hostname]**] [**-help**] [**-version**] [**-j [job_id|job_name|pattern]**] [**-l attr=val,...**] [**-o [owner]**] [**-pe [pe_name]**] [**-q [wc_queue]**] [**-si session_id**] [**-slots [slot_number]**] [**-t task_id_range_list**] [**-P [project_name]**] [**-D [department_name]**] [**-f accounting_file**]

DESCRIPTION

The *qacct* utility scans the accounting data file (see *sge_accounting*(5)) and produces a summary of information for wall-clock time, cpu-time, and system time for the categories of hostname, queue-name, group-name, owner-name, job-name, job-ID and for the queues meeting the resource requirements as specified with the **-l** switch. Combinations of each category are permitted. Alternatively, all or specific jobs can be listed with the **-j** switch. For example the search criteria could include summarizing for a queue and an owner, but not for two queues in the same request.

OPTIONS

-ar [ar_id|ar_name]

The ID or name of the advance reservation for which usage is summarized. If **ar_id** or **ar_name** is not given, accounting data is listed for each advance reservation separately.

-A account_string

The name of the account for which usage is summarized. If **account_string** is not given, accounting data is listed for each owning project separately.

-b begin_time

The earliest start time for jobs to be summarized, in the format [[CC]YY]MMDDhhmm[.SS]. See also **-d** option. If CC is not specified, a YY of < 70 means 20YY.

-d days

The number of days to summarize and print accounting information on. If used together with the **-b begin_time** option (see above), jobs started within **begin_time** to **begin_time** + **days** are counted. If used together with the **-e end_time** (see below) option, count starts at **end_time** - **days**.

-e end_time

The latest start time for jobs to be summarized, in the format [[CC]YY]MMDDhhmm[.SS]. See also **-d** option. If CC is not specified, a YY of < 70 means 20YY.

[-f accounting_file]

The accounting file to be used. If omitted, the system default accounting file is processed.

-g [group_id|group_name]

The numeric system group id or the group alphanumeric name of the job owners to be included in the accounting. If **group_id**/**group_name** is omitted, all groups are accounted.

-h [hostname]

The case-insensitive name of the host upon which accounting information is requested. If the name is omitted, totals for each host are listed separately.

-help

Display help information for the *qacct* command.

-version

Display version information for the *qacct* command.

-j [job_id|job_name|pattern]

The name, a expression for matching names, or ID of the job during execution for which accounting information is printed. If neither a name nor an ID is given all jobs are enlisted. This option changes the output format of *qacct*. If activated, CPU times are no longer accumulated but the "raw" accounting information is printed in a formatted form instead. See *sge_accounting*(5) for an explanation of the displayed information.

-l attr=val,...

A resource requirement specification which must be met by the queues in which the jobs being accounted were executing. The resource request is very similar to the one described in *qsub*(1).

-o [owner]

The name of the owner of the jobs for which accounting statistics are assembled. If the optional **owner** argument is omitted, a listing of the accounting statistics of all job owners being present in the accounting file is produced.

-pe [pe_name]

The name of the parallel environment for which usage is summarized. If **pe_name** is not given, accounting data is listed for each parallel environment separately.

-q [wc_queue_name]

A expression for queues for which usage is summarized. All queue instances matching the expression will be listed. If no expression is specified, a cluster queue summary will be given.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-slots [slot_number]

The number of queue slots for which usage is summarized. If **slot_number** is not given, accounting data is listed for each number of queue slots separately.

-t task_id_range_list

Only available together with the **-j** option described above.

The **-t** switch specifies the array job task range, for which accounting information should be printed. Syntax and semantics of **task_id_range_list** are identical to that one described under the **-t** option to *qsub*(1). Please see there also for further information on array jobs.

-P [project_name]

The name of the project for which usage is summarized. If **project_name** is not given, accounting data is listed for each owning project separately.

-D [department_name]

The name of the department for which usage is summarized. If **department_name** is not given, accounting data is listed for each owning department separately.

-x

Use job end_time instead of start_time for **-b, -e, -d** options

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qacct* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SGE_EXECD_PORT

If set, specifies the tcp port on which *sge_execd*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_execd" instead to define that port.

FILES

<sge_root>/<cell>/common/accounting

Altair Grid Engine default accounting file

<sge_root>/<cell>/common/act_qmaster

Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qsub*(1), *sge_accounting*(5), *sge_session_conf* (5), *sge_qmaster*(8),

COPYRIGHT

qalter

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request "m_mem_free" is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host ("m_mem_free_n<node>").

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule "$pe_slots". Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the "m_mem_free" host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule "$pe_slots"). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can't fulfill the "m_mem_free" request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to "one core on the socket" and decrements the amount of memory on this socket ("m_mem_free_n<nodenumber>"). The global host memory "m_mem_free" on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from "m_mem_free" as well as from "m_mem_free_n<socketnumber>", then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don't offer "m_mem_free" memory are skipped (of course hosts that don't offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory ("m_mem_free_n<N>") and enough free cores, binds the job to these cores and sets memory allocation mode to "cores:strict" (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed ("m_mem_total_n<N>"). Binds the job to the complete socket, decrements the memory on that socket at "m_mem_free_n<N>" (as well as host globally on "m_mem_free") and sets the memory allocation strategy to "cores" (preferred usage of socket local memory).

If the parameters request the "m_mem_free" complex, the resulting NUMA node memory requests can be seen in the "implicit_requests" row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

'n' none (default)
Switches off validation
'e' error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

qconf

NAME

qconf - Altair Grid Engine Queue Configuration

SYNTAX

qconf options

DESCRIPTION

qconf allows the system administrator to add, delete, and modify the current Altair Grid Engine configuration, including queue management, host management, complex management, and user management. *qconf* also allows you to examine the current queue configuration for existing queues.

qconf allows the use of the backslash character ('\') at the end of a line to indicate that the next line is a continuation of the current line. When displaying settings, such as the output of one of the -s* options, *qconf* will break up long lines (lines greater than 80 characters) into smaller lines using backslash line continuation where possible. Lines will only be broken up at commas or whitespace. This feature can be disabled by setting the SGE_SINGLE_LINE environment variable.

OPTIONS

Unless indicated otherwise, the following options and their corresponding operations are available to all users with a valid account.

-Aattr obj_spec fname obj_instance,...

<add to object attributes>

Similar to **-aattr** (see below), but takes specifications for the object attributes to be enhanced from file named **fname**. As opposed to **-aattr**, multiple attributes can be enhanced. Their specifications have to be listed in **fname** according to the file format of the corresponding object (see *sge_queue_conf* (5) for the queue, for example).
Requires root/manager privileges.

-Acal fname|dirname

<add calendar>

Adds a new calendar definition to the Altair Grid Engine environment. Calendars are used in Altair Grid Engine for defining availability and unavailability schedules of queues. The format of a calendar definition is described in *sge_calendar_conf*(5). The calendar definition is taken from the file **fname**. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root/ manager privileges.

-Ackpt fname|dirname

<add ckpt. environment>

Add the checkpointing environment as defined in **fname** (see *sge_checkpoint*(5)) to the list of supported checkpointing environments. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Ace fname|dirname

<add complex>

Add the complex defined in **fname**. The file format of **fname** must comply with the format specified in *sge_complex*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root/manager privileges.

-Aconf fname_list|dirname

<add configurations>

Add the configurations (see *sge_conf*(5)) specified in the files listed in the comma-separated **fname_list**. The configuration is added for the host whose name is identical to the file name. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-adjs session_name

<add a DRMAA2 job session>

Adds a DRMAA2 job session in the master list for the calling user. The DRMAA2 job session can only be used, shown, and deleted by the user who created the session. Users with manager privileges or root can use, show, and delete all DRMAA2 job sessions. A DRMAA2 job session can only be used by DRMAA2 clients, which itself can create and destroy DRMAA2 job sessions.

-adrs session_name

<add a DRMAA2 reservation session>

Adds a DRMAA2 reservation session in the master list for the calling user. The DRMAA2 reservation session can only be used, shown, and deleted by the user who created the session. Users with manager privileges or root can use, show, and delete all DRMAA2 reservation sessions. A DRMAA2 reservation session can only be used by DRMAA2 clients, which itself can create and destroy DRMAA2 reservation sessions.

-Ae fname|dirname

<add execution host>

Add the execution host defined in **fname** to the Altair Grid Engine cluster. The format of the execution host specification is described in *sge_host_conf*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Ahgrp fname|dirname

<add host group configuration>

Add the host group configuration specified in **fname**. The file format of **fname** must comply with the format specified in *sge_hostgroup*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Ajc fname|dirname

<add job class>

Add the job class defined in **fname** to the Altair Grid Engine cluster. The format of the job class specification is described in *sge_job_class*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. This switch can be used by root, managers or users.

-Arqs fname|dirname

<add RQS configuration>

Add the resource quota set (RQS) defined in the file named **fname** to the Altair Grid Engine cluster. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Ap fname|dirname

<add PE configuration>

Add the parallel environment (PE) defined in **fname** to the Altair Grid Engine cluster. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Aprj fname|dirname

<add new project>

Adds the project description defined in **fname** to the list of registered projects (see *sge_project*(5)). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Aq fname|dirname

<add new queue>

Add the queue defined in **fname** to the Altair Grid Engine cluster. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Asi fname

<add new session>

Adds a new session to the Altair Grid Engine environment. Sessions are used in Altair Grid Engine for the synchronization of object state changes within the *sge_qmaster*(8) process to guarantee a consistent view onto the Altair Grid Engine system. Find more information concerning sessions in *sge_session_conf*(5). The format of a session definition is also described in that page.

The session definition is taken from the file **fname**. The unique session ID of the created session will be shown as part of the success message when the qconf*(1)* command returns. This session ID can be used with the **-si** switch of Altair Grid Engine client commands (such as *qstat*(1), *qhost*(1), *qsub*(1) , ...)

Please note that the request to create a new session is automatically done as a transaction within the new session if no other session ID is specified via the **-si** switch. As result all other object state changes that have been done before the session was created will automatically be visible for client commands that use the created session with the corresponding **-si** switch.

To avoid the necessity to specify session parameters and to parse the output of the qconf command, it is possible to use the **-csi** switch of *qconf*(1), which creates a new session with default parameters that prints only the unique session ID to stdout instead of a full message.

Creating sessions requires that the user either has root/manager privileges or is a member of the access control list named **sessionusers**.

-Au fname|dirname

<add an ACL>

Add the user access list (ACL) defined in **fname** to Altair Grid Engine. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. User lists are used for queue usage authentication. Requires root/manager/operator privileges.

-cb

This parameter can be used since Altair Grid Engine version 6.2u5 in combination with the command-line switch **-sep**. In this case the output of the corresponding command will contain information about the added job-to-core binding functionality.

If the **-cb** switch is not used, **-sep** will behave as in GE version 6.2u4 and below.

Please note that this command-line switch will be removed from Altair Grid Engine with the next major release.

-Dattr obj_spec fname obj_instance,...

<del. from object attribs>

Similar to **-dattr** (see below), but the definition of the list attributes from which entries are to be deleted is contained in the file named **fname**. Unlike **-dattr**, multiple attributes can be modified. Their specifications have to be listed in **fname** according to the file format of the corresponding object (see *sge_queue_conf*(5) for the queue, for example).
Requires root/manager privileges.

-Mattr obj_spec fname obj_instance,...

<mod. object attributes>

Similar to **-mattr** (see below) but takes specifications for the object attributes to be modified from the file named **fname**. Unlike **-mattr**, multiple attributes can be modified. Their specifications have to be listed in **fname** according to the file format of the corresponding object (see *sge_queue_conf*(5) for the queue, for example).
Requires root/manager privileges.

-Mc fname

<modify complex>

Overwrites the complex configuration with the contents of **fname**. The argument file must comply with the format specified in *sge_complex*(5). Requires root or manager privilege.

-Mce fname|dirname

<modify complex>

Overwrites the execution host configuration for the specified host with the contents of *fname*, which must comply with the format defined in *sge_complex*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root/manager privileges.

-Mcal fname|dirname

<modify calendar>

Overwrites the calendar definition using information specified in **fname**. The argument file must comply with the format described in *sge_calendar_conf* (5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Mckpt fname|dirname

<modify ckpt. environment>

Overwrite an existing checkpointing environment with the definitions in **fname** (see *sge_checkpoint*(5)). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. The name attribute in **fname** has to match an existing checkpointing environment. Requires root or manager privileges.

-Mconf fname_list|dirname

<modify configurations>

Modify the configurations (see *sge_conf* (5)) as specified in the files listed in the comma-separated **fname_list**. The configuration is modified for the host that is identical to the file name. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Me fname|dirname

<modify execution host>

Overwrites the execution host configuration for the specified host with the contents of **fname**, which must comply with the format defined in *sge_host_conf* (5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Mhgrp fname|dirname

<modify host group configuration>

Allows changing of host group configuration with a single command. All host group configuration entries listed in **fname** will be applied. Configuration entries not listed in **fname** will be deleted. The file format of **fname** must comply with the format specified in *sge_hostgroup*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Mjc fname|dirname

<modify job class configuration>

Allows changing of job class configuration with a single command. All job class configuration entries listed in **fname** will be applied. The file format of **fname** must comply with the format specified in *sge_job_class*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. This switch might be used by managers and owners of the job class that should be modified.

-Mlur fname

<modify license usage records>

Installs a previously-created backup of license usage records for the current cluster. The license usage record database backup will be merged with the currently active database.

-Mrqs fname [mrqs_name]|dirname

<modify RQS configuration>

Same as **-mrqs** (see below) but instead of invoking an editor to modify the RQS configuration, takes the modified configuration from the file **fname**. The name of the rule set in **fname** must be the same as rqs_name. If rqs_name is not set, all rule sets are overwritten by the rule sets in **fname**. Refer to *sge_resource_quota*(5) for details on the RQS configuration format. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Mp fname|dirname

<modify PE configuration>

Same as **-mp** (see below) but instead of invoking an editor to modify the PE configuration, takes the modified configuration from the file **fname**. Refer to *sge_pe*(5) for details on the PE configuration format. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Mprj fname|dirname

<modify project configuration>

Same as **-mprj** (see below) but instead of invoking an editor to modify the project configuration, takes the modified configuration from the file **fname**. Refer to *sge_project*(5) for details on the project configuration format. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Mq fname|dirname

<modify queue configuration>

Same as **-mq** (see below) but instead of invoking an editor to modify the queue configuration, takes the modified configuration from the file **fname**. Refer to *sge_queue_conf*(5) for details on the queue configuration format. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Msi fname

<modify session from file>

Overwrites the session definition as specified in the file **fname**. The argument file must comply with the format described in *sge_session_conf*(5).

Session modifications require that the user either has root/manager privileges or is a member of the access control list named **sessionusers**. The user must be the owner of the session.

-Msconf fname

<modify scheduler configuration from file>

The current scheduler configuration (see *sge_sched_conf*(5)) is overridden with the configuration specified in the file. Requires root or manager privilege.

-Mstree fname

<modify share tree>

Modifies the definition of the share tree (see *sge_share_tree*(5)). The modified sharetree is read from file fname. Requires root or manager privileges.

-Mu fname|dirname

<modify ACL>

Uses the user access list (ACL) defined in **fname** to overwrite any existing ACL with the same name. See *sge_access_list*(5) for information on the ACL configuration format. If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privilege.

-Muser fname|dirname

<modify user>

Modify the user defined in **fname** in the Altair Grid Engine cluster. The format of the user specification is described in *sge_user*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-Rattr obj_spec fname obj_instance,...

<replace object attribs>

Similar to **-rattr** (see below) but the definition of the list attributes whose content is to be replaced is contained in the file named **fname**. Unlike **-rattr**, multiple attributes can be modified. Their specifications have to be listed in **fname** according to the file format of the corresponding object (see *sge_queue_conf*(5) for the queue, for example).
Requires root/manager privileges.

-aattr obj_spec attr_name val obj_instance,...

<add to object attributes>

Allows adding specifications to a single configuration list attribute in multiple instances of an object with a single command. Currently supported objects are the queue, the host, the host group, the parallel environment, the resource quota sets, and the checkpointing interface configuration. These are specified as *queue* , *exechost* , *hostgroup* , *pe* , *resource_quota* or *ckpt* in **obj_spec**. For the obj_spec *queue* the obj_instance can be a cluster queue name, a queue domain name, or a queue instance name. Find more information concerning different queue names in *sge_types*(1). Depending on the type of the obj_instance, this adds to the cluster queues attribute sublist the cluster queues implicit default configuration value or the queue domain configuration value or queue instance configuration value. The queue **load_thresholds** parameter is an example of a list attribute. With the **-aattr** option, entries can be added to such lists. Entries can also be deleted with **-dattr**, modified with **-mattr**, and replaced with **-rattr**.
For the obj_spec *resource_quota* the obj_instance is a unique identifier for a specific rule. The identifier consists of a rule-set name and either the number of the rule in the list, or the name of the rule, separated by a slash ("/").
The name of the configuration attribute to be enhanced is specified with **attr_name** followed by **val** as a *name=value* pair. The comma-separated list of object instances (e.g., the

list of queues) to which the changes are to be applied is specified at the end of the command.
The following restriction applies: For the *exechost* object the **load_values** attribute cannot be modified (see *sge_host_conf*(5)).
Requires root or manager privileges.

-acal calendar_name

<add calendar>

Adds a new calendar definition to the Altair Grid Engine environment. Calendars are used in Altair Grid Engine for defining availability and unavailability schedules of queues. The format of a calendar definition is described in *sge_calendar_conf*(5).

With the calendar name given in the option argument *qconf* will open a temporary file and start up the text editor indicated by the environment variable EDITOR (default editor is *vi*(1) if EDITOR is not set). After entering the calendar definition and closing the editor the new calendar is checked and registered with *sge_qmaster*(8). Requires root/manager privileges.

-ace complex

<add complex>

Adds a new complex with the given name to the list of complexes maintained by Altair Grid Engine. qconf executes *vi*(1) (or \$EDITOR if the EDITOR environment variable is set) to customize a template complex. Upon exit from the editor, the complex is registered with *sge_qmaster*(8). Requires root/manager privileges.

-ackpt ckpt_name

<add ckpt. environment>

Adds a checkpointing environment under the name **ckpt_name** to the list of checkpointing environments maintained by Altair Grid Engine that are usable for submitting checkpointing jobs (see *sge_checkpoint*(5) for details on the format of a checkpointing environment definition). *qconf* retrieves a default checkpointing environment configuration and executes *vi*(1) (or $EDITOR if the EDITOR environment variable is set) to allow you to customize the checkpointing environment configuration. Upon exit from the editor, the checkpointing environment is registered with *sge_qmaster*(8). Requires root/manager privileges.

-aconf host,...

<add configuration>

Successively adds configurations (see *sge_conf*(5)) for the hosts in the comma-separated *fname_list*. For each host, an editor ($EDITOR indicated or *vi*(1)) is invoked and the configuration for the host can be entered. The configuration is registered with *sge_qmaster*(8) after saving the file and quitting the editor.
Requires root or manager privileges.

-ae [host_template]

<add execution host>

Adds a host to the list of Altair Grid Engine execution hosts. If a queue is configured on a host, this host is automatically added to the Altair Grid Engine execution host list. Adding execution hosts explicitly offers the advantage of being able to specify parameters such as load scale values during the registration of the execution host. However, these parameters can be modified (from their defaults) at any later time via the **-me** option described below. If the **host_template** argument is present, *qconf* retrieves the configuration of the specified execution host from *sge_qmaster*(8), or a generic template otherwise. The template is then stored in a file and *qconf* executes *vi*(1) (or the editor indicated by $EDITOR if the EDITOR environment variable is set) to change the entries in the file. The format of the execution host specification is described in *sge_host_conf*(5). When the changes are saved in the editor and the editor is quit the new execution host is registered with *sge_qmaster*(8). Requires root/manager privileges.

-ah hostname,...

<add administrative host>

Adds one or more hosts in **hostname** to the Altair Grid Engine trusted host list (a host must be in this list to execute administrative Altair Grid Engine commands, the sole exception to this being the execution of *qconf* on the *sge_qmaster*(8) node). The default Altair Grid Engine installation procedures usually add all designated execution hosts (see the **-ae** option above) to the Altair Grid Engine trusted host list automatically. Requires root or manager privileges.

-ahgrp group

<add host group configuration>

Adds a new host group with the name specified in **group.** This command invokes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable). The new host group entry is registered after changing the entry and exiting the editor. Requires root or manager privileges.

-ajc jc_name

<add job class configuration>

Adds a new job class with the name specified in **jc_name.** This command invokes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable). The new job class entry is registered after changing the entry and exiting the editor. Job classes can be created by users, root and managers.

-arqs [rqs_name]

<add new RQS>

Adds one or more *Resource Quota Set* (RQS) descriptions under the names **rqs_name** to the list of RQSs maintained by Altair Grid Engine (see *sge_resource_quota*(5) for details on the format of a RQS definition). *qconf* retrieves a default RQS configuration and executes *vi*(1) (or $EDITOR if the EDITOR environment variable is set) to allow you to customize the RQS configuration. Upon exit from the editor, the RQS is registered with *sge_qmaster*(8). Requires root or manager privileges.

-am user|'@'unix_group,...

<add managers>

Adds the indicated users or UNIX groups to the Altair Grid Engine manager list. Requires root or manager privileges.

-ao user|'@'unix_group, ...

<add operators>

Adds the indicated users or UNIX groups to the Altair Grid Engine operator list. Requires root or manager privileges.

-ap pe_name

<add new PE>

Adds a *Parallel Environment* (PE) description with the name **pe_name** to the list of PEs maintained by Altair Grid Engine that are usable for submitting parallel jobs (see *sge_pe*(5) for details on the format of a PE definition). *qconf* retrieves a default PE configuration and executes *vi*(1) (or $EDITOR if the EDITOR environment variable is set) to allow you to customize the PE configuration. Upon exit from the editor, the PE is registered with *sge_qmaster*(8). Requires root/manager privileges.

-at thread_name

<activates thread in master>

Activates an additional thread in the master process. **thread_name** may be **scheduler**, **api**, **jvm**, **lothread** or **reader**. There might be only one scheduler, api, JVM thread, and LO thread. The maximum for reader threads is 64. The corresponding thread is only started when the corresponding maximum is not already reached. **api** threads can only be started if a valid **api_port** is configured in the *sge_bootstrap*(5) file. **reader** threads can only be started if the reader thread pool was enabled during start of qmaster. Initial start of the reader thread pool is enabled by setting the **reader_threads** variable in the *sge_bootstrap*(5) file.

-aprj

<add new project>

Adds a project description to the list of registered projects (see *sge_project*(5)). *qconf* retrieves a template project configuration and executes *vi*(1) (or $EDITOR if the EDITOR environment variable is set) to allow you to customize the new project. Upon exit from the editor, the template is registered with *sge_qmaster*(8). Requires root or manager privileges.

-aq [queue_name]

<add new queue>

qconf retrieves the default queue configuration (see *sge_queue_conf*(5)) and executes *vi*(1) (or $EDITOR if the EDITOR environment variable is set) to allow you to customize the queue configuration. Upon exit from the editor, the queue is registered with *sge_qmaster*(8). A minimal configuration requires only that the queue name and queue hostlist be set. Requires root or manager privileges.

-as hostname,...

<add submit hosts>

Add hosts in **hostname** to the list of hosts from which users are allowed to submit Altair Grid Engine jobs. These are the only hosts from which users are allowed to submit, alter, monitor, and otherwise control jobs. Requires root or manager privileges.

-asi

<add new session>

Adds a new session to the Altair Grid Engine environment. Sessions are used in Altair Grid Engine for the synchronization of object state changes within the *sge_qmaster*(8) process to guarantee a consistent view onto the Altair Grid Engine system. Without using sessions a consistent view of the most recent changes is not guaranteed; in exchange there are better response times for client commands and less load on *sge_qmaster*(8).

Find more information concerning sessions in *sge_session_conf*(5). The format of a session definition is also described in that page.

qconf(1) will open a temporary file and start up the text editor indicated by the environment variable EDITOR (default is vi(1) if EDITOR is not set). After entering the session definition and closing the editor the new session is created and registered with *sge_qmaster*(8).

The unique session ID of the created session will be shown as part of the success message when the *qconf*(1) command returns. This session ID can be used with the **-si** switch of Altair Grid Engine client commands (such as *qstat*(1), *qhost*(1), *qsub*(1) , ...)

Please note that the request to create a new session is automatically done as a transaction within the new session if no other session ID is specified via the **-si** switch. As result all other

object state changes that have been done before the session was created will automatically be visible for client commands that use the created session with the corresponding **-si** switch.

To avoid the necessity to specify session parameters and to parse the output of the *qconf*(1) command, it is possible to use the **-csi** switch of *qconf*(1) which creates a new session with default parameters that prints only the unique session ID to stdout instead of a full message.

Creating sessions requires that the user either has root/manager privileges or is a member of the access control list named **sessionusers**.

-astnode node_path=shares,...

<add share tree node>

Adds the specified share tree node(s) to the share tree (see *sge_share_tree*(5)). The **node_path** is a hierarchical path (**[/]node_name[[/.]node_name...]**) specifying the location of the new node in the share tree. The base name of the node_path is the name of the new node. The node is initialized to the number of specified shares. Requires root or manager privileges.

-astree

<add share tree>

Adds the definition of a share tree to the system (see *sge_share_tree*(5)). A template share tree is retrieved and an editor (either *vi*(1) or the editor indicated by $EDITOR) is invoked for modifying the share tree definition. Upon exiting the editor, the modified data is registered with *sge_qmaster*(8). Requires root or manager privileges.

-Astree fname

<add share tree>

Adds the definition of a share tree to the system (see *sge_share_tree*(5)) from the file **fname**. Requires root or manager privileges.

-au user|'@'unix_group,... acl_name,...

<add users to ACLs>

Adds users or UNIX groups to Altair Grid Engine user access lists (ACLs). User lists are used for queue usage authentication. Requires root/manager/operator privileges.

-Auser fname|dirname

<add user>

Add the user listed in **fname** to the Altair Grid Engine cluster. The format of the user specification is described in *sge_user*(5). If a directory is specified, all files in **dirname** are processed and a bulk operation is performed accordingly. Requires root or manager privileges.

-auser

<add user>

Adds a user to the list of registered users (see *sge_user*(5)). This command invokes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) for a template user. The new user is registered after changing the entry and exiting the editor. Requires root or manager privileges.

-clearusage

<clear sharetree usage>

Clears all user and project usage from the sharetree. All usage will be initialized back to zero.

-clearltusage

<clear long term usage>

Clears all user and long term project usage. All usage will be initialized back to zero.

-cq wc_queue_list

<clean queue>

Cleans queue of jobs which haven't been reaped. Primarily a development tool. Requires root/manager/operator privileges. Find a description of wc_queue_list in *sge_types*(1).

-csi

Adds a new session to the Altair Grid Engine environment. Sessions are used in Altair Grid Engine for the synchronization of object state changes within the *sge_qmaster*(8) process to guarantee a consistent view onto the Altair Grid Engine system. Find more information concerning sessions in *sge_session_conf*(5). The format of a session definition is also described in that page.

The command uses defaults for all session parameters. The executing user of the *qconf*(1) command will be the **owner** of the session. The session duration will be 900 seconds

if not specified otherwise via the **qmaster_param** named **gdi_request_session_timeout** (see *sge_conf*(5)).

On success the command will print the new unique session ID to stdout. Otherwise if creation of the session failed it prints the keyword **NONE**.

Creating sessions requires that the user has root/manager privileges or is a member of the access control list named **sessionusers**.

-dattr obj_spec attr_name val obj_instance,...

<delete in object attribs>
Allows deletion of specifications in a single configuration list attribute in multiple instances of an object with a single command. Find more information concerning obj_spec and obj_instance in the description of **-aattr**

If a specific setting, including all values, should be removed from an object specification, **-purge** has to be used.

-dcal calendar_name,...

<delete calendar>

Deletes the specified calendar definition from Altair Grid Engine. Requires root/manager privileges.

-Dcal fname|dirname

<delete calendar>

Deletes a calendar from file **fname** or from every file in directory **dirname**.

-dckpt ckpt_name

<delete ckpt. environment>

Deletes the specified checkpointing environment. Requires root/manager privileges.

-Dckpt fname|dirname

<delete ckpt. environment>

Deletes a checkpoint from file **fname** or from every file in directory **dirname**.

-dce complex

<delete complex>

Deletes the complex from the Altair Grid Engine complex list. Requires root/manager privileges.

-Dce fname|dirname

<delete complex>

Deletes a complex from file **fname** or from every file in directory **dirname**.

-dconf host,...

<delete local configuration>

The local configuration entries for the specified hosts are deleted from the configuration list. Requires root or manager privilege.

-Dconf fname|dirname

<delete local configuration>

Deletes a configuration from file **fname** or from every file in directory **dirname**.

-ddjs session_name

<delete a DRMAA2 job session>

Deletes a DRMAA2 job session if the session exists and the session was created by the same user. Users with manager privileges or root can delete all DRMAA2 job sessions. Deleting a DRMAA2 session has the same effect as calling *drmaa2_destroy_jsession()* from a DRMAA2 C application.

-ddrs session_name

<delete a DRMAA2 reservation session>

Deletes a DRMAA2 reservation session if the session exists and the session was created by the same user. Users with manager privileges or root can delete all DRMAA2 reservation sessions. Deleting a DRMAA2 session has the same effect as calling *drmaa2_destroy_jsession()* from a DRMAA2 C application.

-de host_name,...

<delete execution host>

Deletes hosts from the Altair Grid Engine execution host list. Requires root or manager privileges.

-De fname|dirname

<delete execution host>

Deletes an execution host from file **fname** or from every file in directory **dirname**.

-dh host_name,...

<*delete administrative host*>

Deletes hosts from the Altair Grid Engine trusted host list. The host on which *sge_qmaster*(8) is currently running cannot be removed from the list of administrative hosts. Requires root or manager privileges.

-dhgrp group

<*delete host group configuration*>

Deletes the host group configuration with the name specified in **group.** Requires root or manager privileges.

-Dhgrp fname|dirname

<*delete host group configuration*>

Deletes a host group from file **fname** or from every file in directory **dirname**.

-djc jc_name

<*delete job class configuration*>

Deletes the job class configuration with the name specified in **jc_name.** Requires root or manager privileges or the deleting user has to be owner of the job class.

-Djc jc_name

<*delete job class configuration*>

Deletes a job class configuration from file **fname** or from every file in directory **dirname**.

-drqs rqs_name_list

<*delete RQS*>

Deletes the specified resource quota sets (RQS). Requires root or manager privileges.

-Drqs fname|dirname

<*delete RQS*>

Deletes a resource quota set from file **fname** or from every file in directory **dirname**.

-dm user|'@'unix_group,...]

<*delete managers*>

Deletes managers or UNIX groups with manager permission from the manager list. Requires root or manager privileges. It is not possible to delete the admin user or the user root from the manager list.

-do user|'@'unix_group,...]

<*delete operators*>

Deletes operators or UNIX groups with operator permissions from the operator list. Requires root or manager privileges. It is not possible to delete the admin user or the user root from the operator list.

-dp pe_name

<*delete parallel environment*>

Deletes the specified parallel environment (PE). Requires root or manager privileges.

-Dp fname|dirname

<*delete parallel environment*>

Deletes a parallel environment from file **fname** or from every file in directory **dirname**.

-dprj project,...

<*delete projects*>

Deletes the specified project(s). Requires root/manager privileges.

-Dprj fname|dirname

<*delete projects*>

Deletes a project from file **fname** or from every file in directory **dirname**.

-dq queue_name,...

<*delete queue*>

Removes the specified queue(s). Active jobs will be allowed to run to completion. Requires root or manager privileges.

-Dq fname|dirname

<delete queue>

Deletes a queue from file **fname** or from every file in directory **dirname**.

-ds host_name,...

<delete submit host>

Deletes hosts from the Altair Grid Engine submit host list. Requires root or manager privileges.

-dsi session_id

<delete session>

Deletes the specified session definition from Altair Grid Engine.

Although sessions are automatically destroyed when the session lifetime ends, it is recommended to delete sessions manually when they are not needed anymore.

Destroying sessions requires that the user has root/manager privileges or is a member of the access control list named **sessionusers**. The user must be the owner of the session.

-dstnode node_path,...

<delete share tree node>

Deletes the specified share tree node(s). The **node_path** is a hierarchical path (**[/]node_name[[/.]node_name...]**) specifying the location of the node to be deleted in the share tree. Requires root or manager privileges.

-dstree

<delete share tree>

Deletes the current share tree. Requires root or manager privileges.

-du user|'@'unix_group,... acl_name,..._<delete users from ACL>_

Deletes one or more users or UNIX groups from one or more Altair Grid Engine user access lists (ACLs). Requires root/manager/operator privileges.

-dul acl_name,...

<delete user lists>

Deletes one or more user lists from the system. Requires root/manager/operator privileges.

-Du fname|dirname

<delete user lists>

Delete a userset from file **fname** or from every file in directory **dirname**.

-duser user,...

<delete users>

Deletes the specified user(s) from the list of registered users. Requires root or manager privileges.

-Duser fname|dirname

<delete users>

Deletes a user from file **fname** or from every file in directory **dirname**.

-help

Prints a listing of all options.

-version

Display version information for the *qconf* command.

-k{m|s|e[j] {host,... |all}}

<shuts down Altair Grid Engine>

Note: The **-ks** switch is deprecated, and may be removed in a future release. Please use the **-kt** switch instead.
Used to shut down Altair Grid Engine components (daemons). When using **-km**, *sge_qmaster*(8) is forced to terminate in a controlled fashion. Similarly, the **-ks** switch causes termination of the scheduler thread. Shutdown of running *sge_execd*(8) processes currently registered is initiated by the **-ke** option. If **-kej** is specified instead, all jobs running on the execution hosts are aborted prior to termination of the corresponding *sge_execd*(8). The comma-separated host list specifies the execution hosts to be addressed by the **-ke** and **-kej** options. If the keyword **all** is specified instead of a host list, all running *sge_execd*(8) processes are shut down. Job abortion initiated by the **-kej** option will result in the **dr** state for all running jobs until *sge_execd*(8) is running again.
Requires root or manager privileges.

-kt thread_name

<terminate master thread>"

Terminates one thread in the master process. Currently it is supported to shut down the **scheduler**, **jvm**, and **lothread** thread. The command will only be successful if the corresponding thread is running. Also one **reader** thread can be terminated with this command if reader thread pool was activated in the *sge_bootstrap*(5) file of qmaster. To shut down more than one reader thread, it is required to execute this command multiple times. Any attempt to shut down the last two reader threads is rejected. They need to remain active so that the *sge_qmaster*(8) process can work properly.

-kec {id,... |all}

<kill event client>

Used to shut down event clients registered at *sge_qmaster*(8). The comma-separated event client list specifies the event clients to be addressed by the **-kec** option. If the keyword **all** is specified instead of an event client list, all running event clients except special clients are terminated.
Special clients are the scheduler thread and the read-only thread pool. Both components provide essential functionality of the *sge_qmaster*(8) component.
Killing the scheduler thread as an event client will automatically terminate the scheduler thread. The thread can be restarted with the *qconf*(1) **-at** command. The read-only thread pool can be shut down as an event client, but the pool will be automatically re-registered. Requires root or manager privilege.

-mattr obj_spec attr_name val obj_instance,...

<modify object attributes>

Allows changing a single configuration attribute in multiple instances of an object with a single command. Find more information concerning obj_spec and obj_instance in the description of **-aattr**

-mc

<modify complex>

The complex configuration (see *sge_complex*(5)) is retrieved, an editor is executed (either *vi*(1) or the editor indicated by $EDITOR) and the changed complex configuration is registered with *sge_qmaster*(8) upon exit of the editor. Requires root or manager privilege.

-mce complex

<modify complex>

Retrieves the current complex configuration for the specified complex, executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the

changed configuration with *sge_qmaster*(8) upon exit from the editor. The format of the complex configuration is described in *sge_complex*(5). Requires root/manager privileges.

-mcal calendar_name

<modify calendar>

The specified calendar definition (see *sge_calendar_conf*(5)) is retrieved, an editor is executed (either *vi*(1) or the editor indicated by $EDITOR) and the changed calendar definition is registered with *sge_qmaster*(8) upon exit of the editor. Requires root or manager privilege.

-mckpt ckpt_name

<modify ckpt. environment>

Retrieves the current configuration for the specified checkpointing environment, executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the new configuration with the *sge_qmaster*(8). Refer to *sge_checkpoint*(5) for details on the checkpointing environment configuration format. Requires root or manager privilege.

-mconf [host,... |global]

<modify configuration>

The configuration for the specified host is retrieved, an editor is executed (either *vi*(1) or the editor indicated by $EDITOR) and the changed configuration is registered with *sge_qmaster*(8) upon exit of the editor. If the optional host argument is omitted or if the special host name **global** is specified, the global configuration is modified. The format of the configuration is described in *sge_conf*(5).
Requires root or manager privilege.

-me hostname

<modify execution host>

Retrieves the current configuration for the specified execution host, executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the changed configuration with *sge_qmaster*(8) upon exit from the editor. The format of the execution host configuration is described in *sge_host_conf*(5). Requires root or manager privilege.

-mhgrp group

<modify host group configuration>

The host group entries for the host group specified in **group** are retrieved and an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) is invoked for modifying the host group configuration. Upon closing the editor, the modified data is registered.

The format of the host group configuration is described in *sge_hostgroup*(5). Requires root or manager privileges.

-mjc jc_name

<modify job class configuration>

The job class entries for the host group specified in **jc_name** are retrieved and an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) is invoked for modifying the job class configuration. Upon closing the editor, the modified data is registered. The format of the job class configuration is described in *sge_job_class*(5). Requires root or manager privileges or the executing user has to be owner of the job class that should be modified.

-mrqs [rqs_name]

<modify RQS configuration>

Retrieves the resource quota set (RQS)configuration defined in rqs_name, or if rqs_name is not given, retrieves all resource quota sets, executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the new configuration with the *sge_qmaster*(8). Refer to *sge_resource_quota*(5) for details on the RQS configuration format. Requires root or manager privilege.

-mp pe_name

<modify PE configuration>

Retrieves the current configuration for the specified *parallel environment* (PE), executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the new configuration with the *sge_qmaster*(8). Refer to *sge_pe*(5) for details on the PE configuration format. Requires root or manager privilege.

-mprj project

<modify project>

Data for the specific project is retrieved (see *sge_project*(5)) and an editor (either *vi*(1) or the editor indicated by $EDITOR) is invoked for modifying the project definition. Upon exiting the editor, the modified data is registered. Requires root or manager privileges.

-mq queuename

<modify queue configuration>

Retrieves the current configuration for the specified queue, executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the new configuration with the *sge_qmaster*(8). Refer to *sge_queue_conf* (5) for details on the queue configuration format. Requires root or manager privilege.

-msi session_id

<modify session>

The specified session definition (see *sge_session_conf*(5) is retrieved, an editor is executed (either vi(1) or the editor indicated by $EDITOR) and the changed session definition is registered with *sge_qmaster*(8) upon exit of the editor.

Session modifications require that the user has root/manager privileges or is a member of the access control list named **sessionusers**. The user must be the owner of the session.

-msconf

<modify scheduler configuration>

The current scheduler configuration (see *sge_sched_conf*(5)) is retrieved, an editor is executed (either *vi*(1) or the editor indicated by $EDITOR) and the changed configuration is registered with *sge_qmaster*(8) upon exit of the editor. Requires root or manager privilege.

-mstnode node_path=shares,...

<modify share tree node>

Modifies the specified share tree node(s) in the share tree (see *sge_share_tree*(5)). The **node_path** is a hierarchical path (**[/]node_name[[/.]node_name...]**) specifying the location of an existing node in the share tree. The node is set to the number of specified **shares**. Requires root or manager privileges.

-mstree

<modify share tree>

Modifies the definition of the share tree (see *sge_share_tree*(5)). The current share tree is retrieved and an editor (either *vi*(1) or the editor indicated by $EDITOR) is invoked for modifying the share tree definition. Upon exiting the editor, the modified data is registered with *sge_qmaster*(8). Requires root or manager privileges.

-mu acl_name

<modify user access lists>

Retrieves the current configuration for the specified user access list, executes an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) and registers the new configuration with the *sge_qmaster*(8). Requires root or manager privilege.

-muser user

<modify user>

Data for the specific user is retrieved (see *sge_user*(5)) and an editor (either *vi*(1) or the editor indicated by the EDITOR environment variable) is invoked for modifying the user definition. Upon exiting the editor, the modified data is registered. Requires root or manager privileges.

-preferred

<send the request to qmaster so that it is handled preferred>

Causes this command to send all requests as priority requests to qmaster so that they will be handled by the priority worker thread pool. Especially in highly loaded clusters, this increases the probability that the requests can be handled earlier than other requests such as qmaster internal requests or requests triggered by other components like execution daemons.

This switch can only be used by managers and it cannot be combined with the -si switch.

-purge queue attr_nm,... obj_spec

<purge divergent attribute settings>

Delete the values of the attributes defined in **attr_nm** from the object defined in **obj_spec**. Obj_spec can be "queue_instance" or "queue_domain". The names of the attributes are described in *sge_queue_conf*(5).
This operation only works on a single queue instance or domain. It cannot be used on a cluster queue. In the case where the **obj_spec** is "queue@@hostgroup", the attribute values defined in **attr_nm** which are set for the indicated hostgroup are deleted, but not those which are set for the hosts contained in that hostgroup. If the **attr_nm** is '*', all attribute values set for the given queue instance or domain are deleted.
The main difference between -dattr and -purge is that -dattr removes a value from a single list attribute, whereas -purge removes one or more overriding attribute settings from a cluster queue configuration. With -purge, the entire attribute is deleted for the given queue instance or queue domain.

-rattr obj_spec attr_name val obj_instance,...

<replace object attributes>

Allows replacement of a single configuration list attribute in multiple instances of an object via a single command. Find more information concerning obj_spec and obj_instance in the description of **-aattr .**
Requires root or manager privilege.

-rsstnode node_path,...

<show share tree node>

Recursively shows the name and shares of the specified share tree node(s) and the names and shares of its child nodes. (See *sge_share_tree*(5).) The **node_path** is a hierarchical path (**[/]node_name[[/.]node_name...]**) specifying the location of a node in the share tree.

-sc

<show complexes>

Displays the complex configuration.

-sce complex

<show complex>

Displays the definition of the specified complex.

-scel

<show complex list>

Displays a list of complex names.

-sceld [ce_list]

<show complex list details>

Shows a detailed list of all complexes or complexes in **ce_list**.

-scal calendar_name

<show calendar>

Displays the configuration of the specified calendar.

-scall

<show calendar list>

Shows a list of all calendars currently defined.

-scalld [cal_list]

<show calendar list details>

Shows a detailed list of all calendars or calendars in **cal_list**.

-sckpt ckpt_name

<show ckpt. environment>

Displays the configuration of the specified checkpointing environment.

-sckptl

<show ckpt. environment list>

Shows a list of the names of all checkpointing environments currently configured.

-sckptld [ckpt_list]

<show ckpt. environment list details>

Shows a detailed list of all checkpoints, or checkpoints in **ckpt_list**.

-scl

<show category list>

Prints information about scheduler categories and jobs that belong to them.
A scheduler category is an automatically managed object in Altair Grid Engine required to optimize Altair Grid Engine internal operations for jobs. Such objects are automatically created for jobs that have the same scheduling characteristics, and they are destroyed when the last job that has those characteristics leaves the system.
One attribute of a scheduler category is a unique character sequence that consists of parts of the job specification. Examples for such parts are resource requests, host and queue selection, and core and memory binding. Which part of a job specification exactly forms a category string depends on various cluster configuration parameters.
This command prints the queue category string, a unique category ID, the number of jobs that belong to the category, and a Boolean attribute named RDI to show whether the corresponding -rdi submit switch (request dispatch information) was used for at least one job that belongs to the category.
Furthermore a full list of job IDs will be shown.

-sconf [host,... |global]

<show configuration>

Print the global or local (host-specific) configuration. If the optional comma-separated host list argument is omitted or the special string **global** is given, the global configuration is displayed. The configuration in effect on a certain host is the merger of the global configuration and the host-specific local configuration. The format of the configuration is described in *sge_conf*(5).

-sconfl

<show configuration list>

Display a list of hosts for which configurations are available. The special host name **global** refers to the global configuration and is not listed. In order to see the global configuration, use **qconf -sconf**.

-sconfld [host_list]

<show configuration list details>

Shows a detailed list of all configurations, or configurations in **host_list**.

-sdjs session_name

<show DRMAA2 job session details>

Shows specific DRMAA2 job sessions detail. Users with manager privileges or root can see DRMAA2 job sessions from all users.

-sdjsl

<show DRMAA2 job session list>

Shows all DRMAA2 job sessions currently defined for the calling user. Users with manager privileges or root can see DRMAA2 job sessions from all users.

-sdrs session_name

<show DRMAA2 reservation session details>

Shows specific DRMAA2 reservation sessions detail. Users with manager privileges or root can see DRMAA2 reservation sessions from all users.

-sdrsl

<show DRMAA2 reservation session list>

Shows all DRMAA2 reservation sessions currently defined for the calling user. Users with manager privileges or root can see DRMAA2 reservation sessions from all users.

-sds

<"show detached settings": show settings that are no longer applicable>

Displays detached settings in the cluster configuration. These are settings which are no longer applicable because of changes in the cluster.

-sdsl

<show a list of all usable DRMAA2 sessions>

Shows specific DRMAA2 reservation sessions detail. Users with manager privileges or root can see DRMAA2 reservation sessions from all users.

-se hostname

<show execution host>

Displays the definition of the specified execution host.

-sel

<show execution hosts>

Displays the Altair Grid Engine execution host list.

-seld [exechost_list]

<show execution hosts details>

Shows a detailed list of all execution hosts, or hosts in **exechost_list**.

-secl

<show event clients>

Displays the Altair Grid Engine event client list. The displayed list columns show the event client ID and the name. The table also contains information about the user who started the event client and from which host the client is connected.

-sec all | {id|name[,id|name]}

<show event client information>

Displays information about Altair Grid Engine event clients. If the keyword **all** is used, the information for all currently registered event clients is shown. Otherwise only the information for the event clients with the specified ID or name is shown. It is possible to mix ID and name specifiers in the argument list.

For “qsub” event clients the **job_id_string** information is available after one event interval has passed. The following event client information is available:

id: Unique event client ID.

name: Name of the event client (e.g. “qsub”, “drmaa”).

host: Host on which the event client is running.

session_key: Internal session key of the event client.

owner: User who registered the event client.

group: Name of the group where the user is a member.

event_interval: Configured event update interval.

buffered_events: Nr. of events pending at the event master. These events are waiting for delivery to the event client.

busy_state: Busy state of the client. If a client is busy the event master will not send events to the client.

register_time: Event client registration time.

busy_alter_time: Latest known time when event client changed the busy state.

latest_ack_time: Time when event master received the latest acknowledge message from the event client.

event_sub_alter_time: Time of last recent event client subscription change.

latest_resync_time: Latest full update time. At startup or on connection errors it is necessary to resync the event client data. This field will be updated in such situations.

job_id_string: Event clients of type "qsub" which are registered when using the qsub -sync option store the job ID string in this field. This information is not available in the first update interval.

-sep

<show licensed processors>

Displays a list of virtual processors. This value is taken from the underlying OS and it depends on the underlying hardware and operating system whether this value represents sockets, cores, or supported threads.

If this option is used in combination with the **-cb** parameter, two additional columns will be shown in the output for the number of sockets and number of cores. Currently SGE will enlist these values only if the corresponding operating system of the execution host is Linux running kernel >= 2.6.16, or Solaris 10. Other operating systems or versions might be supported in future releases. When these values won't be retrieved, the '0' character will be displayed.

-si session_id

<execute within session ID>

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified for the **session_id**, such requests will be executed outside the control of a session.

The attempt to create a session with the **-asi**, **-Asi**, or **-csi** switches will be automatically done within the session that is created if there is no other **session_id** specified with **-si**.

Find more information concerning sessions in *sge_session_conf*(5).

-sh <*show administrative hosts*>

Displays the Altair Grid Engine administrative host list.

-shgrp group

<*show host group config.*>

Displays the host group entries for the group specified in **group.**

-shgrpl

<*show host group lists*>

Displays a name list of all currently defined host groups which have a valid host group configuration.

-shgrpld [hgrp_list]

<*show host group list details*>

Shows a detailed list of all hostgroups, or hostgroups in **hgrp_list**.

-shgrp_tree group

<*show host group tree*>

Shows a treelike structure for the host group.

-shgrp_resolved group

<*show host group hosts*>

Shows a list of all hosts which are part of the definition of the host group. If the host group definition contains sub host groups, these groups are resolved and the hostnames are printed.

-sjc [jc_name]

<*show job class config.*>

Displays the job class parameters specified in **jc_name. Jc_name** might be a job class name or a job class variant name. When the variant part of the name is omitted this command will show the full job class specification; otherwise it will show the settings specific for one variant. If **jc_name** itself is omitted, a template job class is printed.

-sjcl

<show job class lists>

Displays a name list of all currently defined job classes.

-sjcld [jc_list]

<show job class list details>

Shows a detailed list of all job classes, or job classes in **jc_list**.

-srqs [rqs_name_list]

<show RQS configuration>

Shows the definition of the *resource quota sets* (RQS) specified by the argument.

-srqsl

<show RQS-list>

Shows a list of all currently defined *resource quota sets* (RQSs).

-srqsld [rqs_list]

<show RQS-list details>

Shows a detailed list of all resource quota sets, or resource quota sets in **rqs_list**.

-slur

<show license usage records>

Shows a list of all license usage records available for the current cluster.

-sm

<show managers>

Displays the managers list.

-so

<show operators>

Displays the operators list.

-sobjl obj_spec attr_name val

<show object list>

Shows a list of all configuration objects for which val matches at least one configuration value of the attribute whose name matches attr_name.

Obj_spec can be "queue", "queue_domain", "queue_instance", or "exechost". When "queue_domain" is specified, matching is done using the attribute overrides that are specific to the host group. When "queue_instance" is specified, matching is done using the overrides that are specific to the host that is specific to the queue on that host.

Attr_name can be any of the configuration file keywords listed in *sge_queue_conf*(5) or *sge_host_conf*(5). In addition, wildcards can be used to match multiple attributes.

Val can be an arbitrary string or a wildcard expression.

-sp pe_name

<show PE configuration>

Show the definition of the *parallel environment* (PE) specified by the argument.

-spl

<show PE-list>

Show a list of all currently defined *parallel environments* (PEs).

-spld [pe_list]

<show PE-list details>

Shows a detailed list of all parallel environments, or parallel environments in **pe_list**.

-sprj project

<show project>

Shows the definition of the specified project (see *sge_project*(5)).

-sprjl

<show project list>

Shows the list of all currently defined projects.

-sprjld [project_list]

<show project list details>

Shows a detailed list of all projects, or projects in **project_list**.

-sq wc_queue_list

<show queues>

Displays one or multiple cluster queues or queue instances. A description of wc_queue_list can be found in *sge_types*(1).

-sql

<show queue list>

Shows a list of all currently defined cluster queues.

-sqld [queue_list]

<show queue list details>

Shows a detailed list of all queues, or queues in **queue_list**.

-ss

<show submit hosts>

Displays the Altair Grid Engine submit host list.

-ssi session_id

<show session details>

Displays the configuration of the specified session.

-ssil

<show session list>

Displays a list of active sessions including ownership and end time.

-ssconf

<show scheduler configuration>

Displays the current scheduler configuration in the format explained in *sge_sched_conf* (5).

-sstnode node_path,...

<show share tree node>

Shows the name and shares of the specified share tree node(s) (see *sge_share_tree*(5)). The **node_path** is a hierarchical path (**[/]node_name[[/.]node_name...]**) specifying the location of a node in the share tree.

-sstree

<show share tree>

Shows the definition of the share tree (see *sge_share_tree*(5)).

-sst

<show formatted share tree>

Shows the definition of the share tree in a tree view (see *sge_share_tree*(5)).

-sss

<show scheduler status>

Currently displays the host on which the Altair Grid Engine scheduler is active or an error message if no scheduler is running.

-stl

<show thread list>

Shows the names of all activated threads within the *sge_qmaster*(8) process. If multiple threads of one type are running, the corresponding names are suffixed by a unique number. The number of threads that are started during *sge_qmaster*(8) startup are built into qmaster or configured in the bootstrap file. Certain thread types can also be started/stopped dynamically with the **-at**/**-kt** switches of *qconf*(1).

-su acl_name

<show user ACL>

Displays a Altair Grid Engine user access list (ACL).

-sul

<show user lists>

Displays a list of names of all currently defined Altair Grid Engine user access lists (ACLs).

-suld [userset_list]

<show user lists details>

Shows a detailed list of all usersets or usersets in **userset_list**.

-suser user,...

<show user>

Shows the definition of the specified user(s) (see *sge_user*(5)).

-suserl

<show users>

Shows the list of all currently defined users.

-suserld [user_list]

<show users details>

Shows a detailed list of all users or users in **user_list**.

-tlv

<trigger license verification>

Triggers a license verification run.

-tsm

<trigger scheduler monitoring>

Triggers an immediate scheduling run. When the scheduler param **WRITE_SCHEDD_RUNLOG** is set, the Altair Grid Engine scheduler is forced by this option to print trace messages of its next scheduling run to the file <sge_root>/<cell>/common/schedd_runlog_. The messages indicate the reasons for jobs and queues not being selected in that run. Requires root or manager privileges.

Note that the reasons for job requirements being invalid with respect to resource availability of queues are displayed using the format as described for the *qstat*(1) **-F** option (see description of **Full Format** in section **OUTPUT FORMATS** of the *qstat*(1) manual page.

-uha

<trigger host_aliases file re-read>

Triggers an immediate re-read of the host_aliases file (<sge_root>/<cell>/common/host_aliases). This option can be used to force the *sge_qmaster*(8) to adopt changes made in that file. The default behavior is that qmaster checks the status of the host_aliases file every 60 seconds. It is possible to disable this check by setting the qmaster_params option DISABLE_HOST_ALIASES_CHECK (see *sge_conf*(5) man page). By doing this the host_aliases file is only re-read when triggered via the *qconf -uha* option.

The *qconf -uha* option can be used to synchronize the running qmaster with latest changes in the host_aliases file. In this case it is recommended to use the DISABLE_HOST_ALIASES_CHECK. Once the changes to the host_aliases file are done the return value of *qconf -uha* has the following meanings:

exit_status	description
0	qmaster did re-read the host_aliases file and there was no error
1	qmaster did not see any changes (e.g. the modification time of the host_aliases file did not change) - retry required
2	qmaster could not obey all changes and must be restarted

The output line of *qconf -uha* contains extra information which is explained in the following table:

value	description
add	Number of added host aliases
del	Nunber of deleted host aliases
skipped	Number of host aliases that were skipped because a qmaster restart is needed
timestamp	Modification time of the last host_aliases file that was parsed by qmaster. A value of 1 means that there was never a host_aliases file available. The value 2 means that there was a host_aliases file active at startup, but later it was deleted.
size	The size of the parsed host_aliases file

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address the Altair Grid Engine cell, *qconf* uses (in order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the TCP port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry to define that port.

SGE_EXECD_PORT

If set, specifies the TCP port on which *sge_execd*(8) is expected to listen for communication requests. Most installations will instead use a services map entry to define that port.

SGE_SINGLE_LINE

If set, indicates that long lines should not be broken up using backslashes. This setting is useful for scripts that expect one entry per line.

RESTRICTIONS

Modifications to a queue configuration do not affect an active queue, instead taking effect on the next invocation of the queue (i.e., the next job).

FILES

<sge_root>/<cell>/common/act_qmaster

Altair Grid Engine master host file

SEE ALSO

sge_intro(1), qstat*(1)*, *sge_checkpoint*(5),*sge_complex*(5), *sge_conf*(5), *sge_host_conf*(5), *sge_pe*(5), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_execd*(8), *sge_qmaster*(8), *sge_resource_quota*(5)

COPYRIGHT

qdel

NAME

qdel - delete Altair Grid Engine jobs from queues

SYNTAX

qdel [**-f**] [**-help**] [**-u wc_user_list**] [**wc_job_range_list**] [**-si session_id**] [**-t task_id_range**]

DESCRIPTION

Qdel provides a means for a user/operator/manager to delete one or more jobs. A manager/operator can delete jobs belonging to any user, while a regular user can only delete his or her own jobs. If a manager wants to delete another user's job, the manager can specify the job id. If the manager is using a job name or pattern, he or she must also specify the user's name via "-u wc_user_list". A "qdel wc_job_name" will delete only the jobs of the calling user by default. *Qdel* deletes jobs in the order in which their job identifiers are presented. Find additional information concerning *wc_user_list* and *wc_job_list* in *sge_types*(1).

OPTIONS

-f

Force deletion of job(s). The job(s) are deleted from the list of jobs registered at sge_qmaster(8) even if the sge_execd(8) controlling the job(s) does not respond to the delete request sent by *sge_qmaster*(8).

Users which are neither Altair Grid Engine managers nor operators can only use the **-f** option (for their own jobs) if the cluster configuration entry **qmaster_params** contains the flag ENABLE_FORCED_QDEL (*sge_conf*(5)). However, behavior for administrative and non-administrative users differs. Jobs are deleted from the Altair Grid Engine database immediately in case of administrators. Otherwise, a regular deletion is attempted first and a forced cancellation is only executed if the regular deletion was unsuccessful.

Additionally regular qdel requests can result in a forced deletion of a job if **ENABLE_FORCED_QDEL_IF_UNKNOWN** is set in the **qmaster_params** (see *sge_conf*(5))

-help

Prints a listing of all options.

-version

Display version information for the *qdel* command.

-s {p|r|s|S|N|P|hu|ho|hs|hd|hj|ha|h|a}[+]

Deletes only jobs in the specified state, any combination of states is possible. To delete jobs in user/operator/system/array-dependency hold, use the -s hu/ho/hs/hd option. To delete preempted jobs in the S, N or P state use the -s S/N/P option. The -s ha option deletes jobs which where submitted with the qsub -a command. qdel -s hj deletes all jobs which are not eligible for execution unless the job has entries in the job dependency list.qdel -s h is an abbreviation for qdel -s huhohshdhjha and qdel -s a is an abbreviation for qdel -s psr (see -a, -hold_jid and -hold_jid_ad options to qsub(1)). qdel -s p deletes jobs in pending state and in any hold state.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id* then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-t

The -t switch can be passed as additional switch after the specification of a job name or job ID representing an array job. In this case not the full array job will be deleted but only the specified tasks. The command “qdel 137 -t 5-9:2 work -t 2-3” will delete those tasks with uneven tasks IDs from 5 to 9 from the array job 137 and it will also remove the tasks 2 and 3 from all array jobs where the name is ‘work’. “qdel 137.5-9:2” is an equivalent for the first part of the deletion command above. This notation can not be used in combination with jobs names.

-u wc_user_list

Deletes only those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use *qdel -u "*" to delete all jobs of all users. If a manager wants to delete a specific job of a user, he has to specify the user and the job. If no job is specified all jobs from that user are deleted.

wc_job_range_list

A list of jobs, which should be deleted

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qdel* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

FILES

<SGE_ROOT>/<SGE_CELL>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qstat*(1), *qsub*(1), *sge_session_conf*(5), *sge_qmaster*(8), *sge_execd*(8).

COPYRIGHT

qhold

NAME

qhold - hold back Altair Grid Engine jobs from execution

SYNTAX

qhold [**-h** {**u**|**o**|**s**},...] [**-help**] [**-si session_id**] [**job/task_id_list**]

qhold [**-h** {**u**|**o**|**s**},...] [**-help**] [**-si session_id**] **-u user_list**

DESCRIPTION

Qhold provides a means for a user/operator/manager to place so called *holds* on one or more jobs pending to be scheduled for execution. As long as any type of hold is assigned to a job, the job is not eligible for scheduling.

Holds can be removed with the *qrls*(1) or the *qalter*(1) command.

There are three different types of holds:

user User holds can be assigned and removed by managers, operators and the owner of the jobs.

operator Operator holds can be assigned and removed by managers and operators.

system System holds can be assigned and removed by managers only.

If no hold type is specified with the **-h** option (see below) the user hold is assumed by default.

An alternate way to assign holds to jobs is the *qsub*(1) or the *qalter*(1) command (see the **-h** option).

OPTIONS

-h {u|o|s},...

Assign a u(ser), o(perator) or s(system) hold or a combination thereof to one or more jobs.

-help

Prints a listing of all options.

-version

Display version information for the *qhold* command.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session. Find more information concerning sessions in *sge_session_conf*(5).

-u username,...

Changes are only made on those jobs which were submitted by users specified in the list of usernames. Managers are allowed to use the **qhold -u "*"** command to set a hold for all jobs of all users. If a user uses the **-u** switch, the user may specify an additional **job/task_id_list**.

job/task_id_list

Specified by the following form:

```
job_id[.task_range][,job_id[.task_range],...]
```

If present, the *task_range* restricts the effect of the *qhold* operation to the array job task range specified as suffix to the job id (see the **-t** option to *qsub*(1) for further details on array jobs).

The task range specifier has the form n[-m[:s]]. The range may be a single number, a simple range of the form n-m or a range with a step size.

Instead of *job/task_id_list* it is possible to use the keyword 'all' to modify the hold state for all jobs of the current user.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qhold* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SEE ALSO

sge_intro(1), *sge_session_conf*(5), *qalter*(1), *qrls*(1), *qsub*(1).

COPYRIGHT

qhost

NAME

qhost - show the status of Altair Grid Engine hosts, queues, jobs

SYNTAX

qhost [**-F** [**resource_name,...**]] [**-help**] [**-h host_list**] [**-j**] [**-l resource=val,...**] [**-ncb**] [**-si session_id**] [**-st**] [**-u user,...**] [**-xml**]

DESCRIPTION

qhost shows the current status of the available Altair Grid Engine hosts, queues and the jobs associated with the queues. Selection options allow you to get information about specific hosts, queues, jobs or users. If multiple selections are done a host is only displayed if all selection criteria for a host are met. Without any options *qhost* will display a list of all hosts without queue or job information.

OPTIONS

-F [resource_name,...]

qhost will present a detailed listing of the current resource availability per host with respect to all resources (if the option argument is omitted) or with respect to those resources contained in the resource_name list. Please refer to the description of the **Full Format** in section **OUTPUT FORMATS** below for further detail.

-help

Prints a listing of all options.

-version

Display version information for the *qhost* command.

-h host_list

Prints a list of all hosts contained in host_list.

-j

Prints all jobs running on the queues hosted by the shown hosts. This switch calls **-q** implicitly.

-json

This option can be used with all other options and changes the output to JSON. The used schemas are referenced in the JSON output. The output is printed to stdout.

-l resource[=value],...

Defines the resources to be granted by the hosts which should be included in the host list output. Matching is performed on hosts based on non-mutable resource availability information only. That means load values are always ignored except the so-called static load values (i.e. “arch”, “num_proc”, “mem_total”, “swap_total” and “virtual_total”) ones. Also consumable utilization is ignored. If there are multiple -l resource requests they will be concatenated by a logical AND: a host needs to match all resources to be displayed.

-ncb

This command line switch can be used in order to get 6.2u5 compatible output with other *qhost*(1) command line switches. In that case the output of the corresponding command will suppress information concerning the execution host topology. Note that this option will be removed in the next major version.

-q

Show information about the queues instances hosted by the displayed hosts.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-st

If **-F** is used, not only the currently free resources are shown, but also the total amount, e.g. **gc:lic=5/20**, where 5 licenses remain from a total of 20.

-u user,...

Display information only on those jobs and queues being associated with the users from the given user list.

-xml

This option can be used with all other options and changes the output to XML. The used schemas are referenced in the XML output. The output is printed to stdout.

If the **-xml** parameter is combined with **-ncb** then the XML output will contain 6.2u5 compatible output.

OUTPUT FORMATS

Depending on the presence or absence of the **-q** or **-F** and **-j** option three output formats need to be differentiated.

Default Format (without -q, -F and -j)

For each host one line is printed. The output consists of consisting of

- the Hostname
- the Architecture.
- the Number of processors.
- the Load.
- the Total Memory.
- the Used Memory.
- the Total Swapspace.
- the Used Swapspace.

If the **-q** option is supplied, each host status line also contains extra lines for every queue hosted by the host consisting of,

- the queue name.

- the queue type - one of B(atch), I(nteractive), C(heckpointing), P(arallel) or combinations thereof,
- the number of reserved, used and available job slots,
- the state of the queue - one of u(nknown) if the corresponding *sge_execd*(8) cannot be contacted, a(larm), A(larm), B(locked on SSOS), C(alendar suspended), s(uspended), S(ubordinate), d(isabled), D(isabled), E(rror) or combinations thereof.

If the state is a(alarm) at least one of the load thresholds defined in the *load_thresholds* list of the queue configuration (see *sge_queue_conf*(5)) is currently exceeded, which prevents from scheduling further jobs to that queue.

As opposed to this, the state A(larm) indicates that at least one of the suspend thresholds of the queue (see *sge_queue_conf*(5)) is currently exceeded. This will result in jobs running in that queue being successively suspended until no threshold is violated.

The states s(uspended) and d(isabled) can be assigned to queues and released via the *qmod*(1) command. Suspending a queue will cause all jobs executing in that queue to be suspended.

The states D(isabled) and C(alendar suspended) indicate that the queue has been disabled or suspended automatically via the calendar facility of Altair Grid Engine (see *sge_calendar_conf*(5)), while the S(ubordinate) state indicates, that the queue has been suspend via subordination to another queue (see *sge_queue_conf*(5) for details). When suspending a queue (regardless of the cause) all jobs executing in that queue are suspended too.

The state B(locked on SSOS) indicates that this subordinate queue is blocked from running any new jobs as the threshold for the combined number of jobs on the parent and child queues has been reached.

If an E(rror) state is displayed for a queue, *sge_execd*(8) on that host was unable to locate the *sge_shepherd*(8) executable on that host in order to start a job. Please check the error logfile of that *sge_execd*(8) for leads on how to resolve the problem. Please enable the queue afterwards via the **-c** option of the *qmod*(1) command manually.

If the **-F** option was used, resource availability information is printed following the host status line. For each resource (as selected in an option argument to **-F** or for all resources if the option argument was omitted) a single line is displayed with the following format:

*a one letter specifier indicating whether the current resource availability value was dominated by either `**g**' - a cluster global or`**h**' - a host total

*a second one letter specifier indicating the source for the current resource availability value, being one of `**l**' - a load value reported for the resource,`**L**' - a load value for the resource after administrator defined load scaling has been applied, `**c**' - availability derived from the consumable resources facility (see *complexes*(5)),`**f**' - a fixed availability definition derived from a non-consumable complex attribute or a fixed resource limit.

- after a colon the name of the resource on which information is displayed.
- after an equal sign the current resource availability value.

- Optional: If **-st** was used as well, after the current value follows a slash ("/") and the total resource amount.

The displayed availability values and the sources from which they derive are always the minimum values of all possible combinations. Hence, for example, a line of the form "qf:h_vmem=4G" indicates that a queue currently has a maximum availability in virtual memory of 4 Gigabyte, where this value is a fixed value (e.g. a resource limit in the queue configuration) and it is queue dominated, i.e. the host in total may have more virtual memory available than this, but the queue doesn't allow for more. Contrarily a line "hl:h_vmem=4G" would also indicate an upper bound of 4 Gigabyte virtual memory availability, but the limit would be derived from a load value currently reported for the host. So while the queue might allow for jobs with higher virtual memory requirements, the host on which this particular queue resides currently only has 4 Gigabyte available.

If there are additional resources that are available through preempted jobs then the resource amount is shown in the form "+1". This additional output is optional.

If the option **-st** is used, the total amount of a given resource is appended in the form of "/20"

If positive or negative affinity is defined for one or more resources that are consumed by jobs running on a hosts or queue then the corresponding affinity values will be shown in the form "(haff=1.000000)" additionally. haff shows the internal host affinity values whereas qaff shows the queue affinity values.

A string like "hc:gpu=2+1/4 (haff=5.000000)" shows the amount of resources of the gpu consumable. In total there are 4, currently are 3 available (one of those gpus due to a preempted job). Host affinity for gpu is 5.0.

After the queue status line (in case of **-j**) a single line is printed for each job running currently in this queue. Each job status line contains

- the job ID,
- the job name,
- the job owner name,
- the status of the job - one of t(ransfering), r(unning), R(estarted), s(uspended), S(uspended) or T(hreshold) (see the **Reduced Format** section for detailed information),
- the start date and time and the function of the job (MASTER or SLAVE - only meaningful in case of a parallel job) and
- the priority of the jobs.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qhost* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SGE_GDI_REQUEST_REDUCE_LEVEL

This environment variable can be used to reduce the amount of data transferred to the qhost client. The qhost client is requesting needed data lists from the sge_qmaster daemon in order to provide the requested output. It is possible to reduce the data requested by selecting different reduction levels.

The default value for this parameter is "1". If the environment variable is not set at all the resulting reduction level is also "1".

The level "0" can be used to turn off reduction completely which will result in larger data sizes requested from the sge_qmaster daemon.

The level "2" will reduce more data than level "1" but is currently in "experimental" state. If an error occurs there would be a critical logging and qhost will terminate with a non zero exit state.

FILES

SGE_ROOT/SGE_CELL/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), qalter(1), qconf(1), qhold(1), qmod(1), qsub(1), sge_queue_conf(5), sge_session_conf(5), sge_execd(8), sge_qmaster(8), sge_shepherd(8)

COPYRIGHT

qlogin

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request "m_mem_free" is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host ("m_mem_free_n<node>").

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule "$pe_slots". Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the "m_mem_free" host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule "$pe_slots"). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can't fulfill the "m_mem_free" request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to "one core on the socket" and decrements the amount of memory on this socket ("m_mem_free_n<nodenumber>"). The global host memory "m_mem_free" on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from "m_mem_free" as well as from "m_mem_free_n<socketnumber>", then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don't offer "m_mem_free" memory are skipped (of course hosts that don't offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory ("m_mem_free_n<N>") and enough free cores, binds the job to these cores and sets memory allocation mode to "cores:strict" (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed ("m_mem_total_n<N>"). Binds the job to the complete socket, decrements the memory on that socket at "m_mem_free_n<N>" (as well as host globally on "m_mem_free") and sets the memory allocation strategy to "cores" (preferred usage of socket local memory).

If the parameters request the "m_mem_free" complex, the resulting NUMA node memory requests can be seen in the "implicit_requests" row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

'n' none (default)
Switches off validation
'e' error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

qmake

NAME

qmake - distributed parallel make, scheduling by Altair Grid Engine.

SYNTAX

qmake [**options**] – [**gmake options**]

DESCRIPTION

Qmake is a parallel, distributed *make*(1) utility. Scheduling of the parallel *make*(1) tasks is done by Altair Grid Engine. It is based on *gmake* (GNU make), version 4.0. Both Altair Grid Engine and *gmake* command line options can be specified. They are separated by "–".

All Altair Grid Engine options valid with *qsub*(1) or *qrsh*(1) can be specified with *qmake* - see *submit*(1) for a description of all Altair Grid Engine command line options. The *make*(1) manual page describes the *gmake* command line syntax.

The syntax of *qmake* makefiles corresponds to *gmake* and is described in the "GNU Make Manual".

A typical qmake call will use the Altair Grid Engine command line options -cwd to have a scheduled make started in the current working directory on the execution host, -v PATH if the Altair Grid Engine environment is not setup in the users .cshrc or .profile shell resource file and request slots in a parallel environment (see *sge_pe*(5)).

If no resource request (Altair Grid Engine command line option -l) is specified, qmake will use the environment variable SGE_ARCH to request the same architecture for task execution as has the submit host. If SGE_ARCH is set, the architecture specified in SGE_ARCH will be requested by inserting the option -l arch=$SGE_ARCH into the command line options. If SGE_ARCH is not set, the make tasks can be executed on any available architecture. As this is critical for typical make (compile) jobs, a warning will be output.

qmake has two different modes for allocating Altair Grid Engine resources for the parallel execution of tasks:

1. Allocation of resources using a parallel environment. If the -pe option is used on the qmake command line, a parallel job is scheduled by Altair Grid Engine. The make rules are executed as tasks within this parallel job.
2. Dynamic allocation of resources. If no parallel environment is requested when submitting a qmake job, each make rule will generate an individual Altair Grid Engine qrsh

job. All resource requests given to qmake will be inherited by the jobs processing the make rules.

In dynamic allocation mode, additional resource requests for individual rules can be specified by preceding the rule by the definition of an environment variable SGE_RREQ. The rule then takes the form SGE_RREQ="request" rule, e.g. SGE_RREQ="-l lic=1" cc -c ... If such makefile rules are executed in a make utility other than qmake, the environment variable SGE_RREQ will be set in the environment established for the rule's execution - without any effect.

EXAMPLES

```
qmake -cwd -v PATH -pe compiling 1-10 --
```

will request between 1 and 10 slots in parallel environment "compiling". If the SGE_ARCH environment variable is set to the machines architecture, a resource request will be inserted into the qmake command line to start the qmake job on the same architecture as the submit host. The *make* tasks will inherit the complete environment of the calling shell. It will execute as many parallel tasks as slots have been granted by Altair Grid Engine.

```
qmake -l arch=sol-sparc -cwd -v PATH -- -j 4
```

will submit each make rule as an individual qrsh job. A maximum of 4 tasks will be processed in parallel. The qmake job will be started on a machine of architecture sol-sparc, this resource request will also be inherited by the make tasks, i.e. all jobs created for the execution of make tasks will request the architecture sol-sparc.

If the following Makefile is submitted with the above command line, additional resource requests will be made for individual rules: For the compile and link rules, compiler licenses (comp) and linker licenses (link) will be requested, in addition to the resource request made for the whole job (-l arch=sol-sparc) on the command line.

```
all: test

clean:
     rm -f test main.o functions.o

test: main.o functions.o
     SGE_RREQ="-l link=1" ld -o test main.o functions.o

main.o: main.c
     SGE_RREQ="-l comp=1" cc -c -DALIASPATH=\"/usr/local/share/locale:.\" -o main.o main.c

functions.o: functions.c
     SGE_RREQ="-l comp=1" cc -c -DALIASPATH=\"/usr/local/share/locale:.\" -o functions.o func
```

The command line

```
qmake -cwd -v PATH -l arch=sol-sparc64 -pe make 3 --
```

will request 3 parallel *make* tasks to be executed on hosts of architecture “sol-sparc64”. The submit may be done on a host of any architecture.

The shell script

```
#!/bin/sh
qmake -inherit --
```

can be submitted by

```
qsub -cwd -v PATH -pe make 1-10 [further sge options] <script>
```

Qmake will inherit the resources granted for the job submitted above under parallel environment “make”.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qmake* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_ARCH

The architecture of the submit host. If this variable is set in the submission environment, qmake will request the given architecture for job execution (see DESCRIPTION above).

KNOWN PROBLEMS

Slow NFS server

Very low file server performance may lead to problems on depending files.

Example: Host a compiles a.c to a.o, host b compiles b.c to b.o, host c shall link program c from a.o and b.o. In case of very bad NFS performance, host c might not yet see files a.o and b.o.

Multiple commands in one rule

If multiple commands are executed in one rule, the makefile has to ensure that they are handled as one command line.

Example:

```
libx.a:
        cd x
        ar ru libx.a x.o
```

Building libx.a will fail, if the commands are executed in parallel (and possibly on different hosts). Write the following instead:

```
libx.a:
        cd x ; ar ru libx.a x.o
```

or

```
libx.a:
        cd x ; \\
        ar ru libx.a x.o
```

SEE ALSO

submit(1), *sge_pe*(5), as well as *make*(1) (GNU make manpage) and *The GNU Make Manual* in SGE_ROOT/3rd_party/qmake.

COPYRIGHT

Qmake contains portions of Gnu Make (*gmake*), which is the copyright of the Free Software Foundation, Inc., Boston, MA, and is protected by the Gnu General Public License.

See *sge_intro*(1) and the information provided in SGE_ROOT/3rd_party/qmake for a statement of further rights and permissions.

qmod

NAME

qmod - modify a Altair Grid Engine queue, running job or job class

SYNTAX

qmod [**options**]

DESCRIPTION

Qmod enabled users classified as owners of a running job, queue or job class to modify the state of that object. Users that have submitted a job automatically own the job. Queue and job class ownership is defined by the *owner* attribute within a queue and by the *owner_list* attribute defined in the job class (see *sge_queue_conf*(5) or *sge_job_class*(5) for details).

A manager/operator or root can execute *qmod* for any object in a Altair Grid Engine but only from administrative hosts.

The switches below expect the arguments *wc_queue_list*, *wc_job_list* or *jc_list* that describe the object that should be modified. Find the description for these arguments in *sge_types*(1).

OPTIONS

-c wc_job_range_list | wc_queue_list

Note: Deprecated, may be removed in future release. Please use the **-cj** or **-cq** switch instead.

Clears the error state of the specified jobs(s)/queue(s).

-cj wc_job_range_list

Clears the error state of the specified jobs(s).

-cq wc_queue_list

Clears the error state of the specified queue(s).

-d wc_queue_list

Disables the queue(s), i.e. no further jobs are dispatched to disabled queues while jobs already executing in these queues are allowed to finish.

-djc jc_list

Disables the job class variant(s) that are specified. If only the job class name is specified (without variant name) then only the default variant of a job class will be disabled whereas all other variants remain in their previous state. Job class variants that are in the disabled state cannot be used to create new jobs.

-e wc_queue_list

Enables the queue(s).

-E wc_queue_list

Sets queue(s) into error state.

-ejc jc_list

Enables the job class variant(s) that are specified. If only the job class name is specified (without variant name) then only the default variant of a job class will be disabled whereas all other variants remain in their previous state. Job class variants that are in enabled state can be used to create new jobs.

-f

Force the modification action for the queue despite the apparent current state of the queue. For example if a queue appears to be suspended but the job execution seems to be continuing the manager/operator can force a suspend operation which will send a SIGSTOP to the jobs. In any case, the queue or job status will be set even if the sge_execd(8) controlling the queues/jobs cannot be reached. Requires manager/operator privileges.

-help

Prints a listing of all options.

-version

Display version information for the *qmod* command.

-msg [message]

The **-msg** switch can be used in combination with other switches of this command that trigger queue state changes. The specified message will be attached to specified queues. If message is omitted then previously attached messages will be deleted. The message string will be displayed by qstat if the **-explain m** switch is specified for this command.

-preferred

Causes this command to send all requests as priority requests to qmaster so that they will be handled by the priority worker thread pool. Especially in highly loaded cluster this increases the probability that the requests can be handled earlier in comparison to other requests (e.g. qmaster internal requests or requests triggered by other components like execution daemons).

The switch can only be used by managers and it cannot be combined with the -si switch.

-p jid wc_job_task_list [preemptive_action]

Sends a manual preemption request to the Altair Grid Engine system.

The job specified by *jid* has to be pending and it will be seen as preemptor candidate by the Altair Grid Engine system. All other jobs (or tasks of an array job) specified by *wc_job_task_list* have to be running and they act as preemptee candidates.

The Altair Grid Engine scheduler tries to start the preemptor candidate if there are no other jobs with a higher effective priority. If there are no free resources available then the system will preempt the preemptee candidates to gain access to the used resources so that the preemptor candidate can be started.

The method how preemptee candidates are preempted can be specified by the *preemption_action*. It can be one of the letters (**S**)suspend, e(**N**)handed suspend, (**P**)reempt, (**C**)heckpoint, (**R**)erun or (**T**)erminate. If the specified preemption action is not available for a preemptee candidate or when the **preemption_action** is not specified then the system might choose an appropriate preemption action to gain required resources.

For more detail about preemption and the differences of the preemptive actions read *sge_preemption*(5)

-r wc_job_range_list | wc_queue_list

Note: Deprecated, may be removed in future release. Please use the **-rj** or **-rq** switch instead.

If applied to queues, reschedules all jobs currently running in this queue. If applied to running jobs, reschedules the jobs. Requires root or manager privileges. In order for a job to be rescheduled, it or the queue in which it is executing must have the rerun flag activated. (See **-r** option in the *qsub*(1) man page and the *rerun* option in the *sge_queue_conf*(5) man page for more information.)

Additional restrictions apply for parallel and checkpointing jobs. (See the *reschedule_unknown* description in the *sge_conf*(5) man page for details).

-rj wc_job_range_list

If applied to running jobs, reschedules the jobs if they can be rescheduled. See also the **-r** switch. Requires root or manager privileges.

-rq wc_queue_list

If applied to queues, reschedules all jobs currently running in this queue. For parallel jobs, the job is rescheduled only if the master task of this parallel job is running in this queue. See also the **-r** switch. Requires root or manager privileges.

-s wc_job_range_list | wc_queue_list

Note: Deprecated, may be removed in future release. Please use the **-sj** or **-sq** switch instead.

If applied to queues, suspends the queues and any jobs which might be active. If applied to running jobs, suspends the jobs.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-sj wc_job_range_list

If applied to running jobs, suspends the jobs. For parallel jobs, Altair Grid Engine sends the STOP signal to the master task only. The master task has to handle the suspending of all slave tasks itself then. This is because Altair Grid Engine doesn't know in which order the slave tasks have to be suspended. If a job is both suspended explicitly and via suspension of its queue, a following unsuspend of the queue will not release the suspension state on the job.

-sq wc_queue_list

If applied to queues, suspends the queues and any jobs which might be active. For parallel jobs, Altair Grid Engine sends the STOP signal to the master task only (see **-sj** switch). If any slave task of a parallel job runs in a queue that is to be suspended, the master task of this job is suspended, no matter where it is running.

-us wc_job_range_list | wc_queue_list

Note: Deprecated, may be removed in future release. Please use the **-usj** or **-usq** switch instead.

If applied to queues, unsuspends the queues and any jobs which might be active. If applied to jobs, unsuspends the jobs. For parallel jobs, please see the **-usj** or **-usq** switch.

-usj wc_job_range_list

If applied to jobs, unsuspends the jobs. For parallel jobs, Altair Grid Engine sends the CONT signal to the master task only. The master task has to handle the unsuspending of all slave tasks itself then. If a job is both suspended explicitly and via suspension of its queue, a following unsuspend of the queue will not release the suspension state on the job. A parallel job is not un-suspended as long as at least one task of this job is suspended for any reason.

-usq wc_queue_list

If applied to queues, unsuspends the queues and any jobs which might be active. For parallel jobs, Altair Grid Engine sends the CONT signal to the master task only (see **-usj** switch). A parallel job is not unsuspended as long as at least one task of this job is suspended for any reason.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qmod* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

FILES

SGE_ROOT/SGE_CELL/common/act_qmaster
Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *sge_ckpt*(1), *qstat*(1), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_preemption*(5), *sge_execd*(8), *sge_types*(1), *sge_job_class*(5).

COPYRIGHT

qmon

NAME

qmon - X-Windows OSF/Motif graphical user's interface for Altair Grid Engine

SYNTAX

qmon [options]

DESCRIPTION

Qmon allows administrators and users to manipulate the Altair Grid Engine system from an X-Window environment. *Qmon* provides various dialogues linked together in multiple ways. For each task the user wishes to accomplish via *qmon* a corresponding dialogue is provided. There are multiple ways to address the proper dialogue for a certain task:

- The *qmon* main window that comes up first on start-up contains icon buttons for all major administrative and user tasks. A functionality tooltip is displayed when pointing at the different icons.
- A **Task** pulldown menu button appears in the *qmon* main window menu bar. Clicking on it opens a list of available tasks. Selecting one of them opens the corresponding dialogue.
- The **Task** pulldown menu also contains the key accelerators which can be used to invoke the task dialogues directly from the main window by pressing a certain button sequence on the keyboard.
- While navigating through a certain dialogue and its dialogue subhierarchy, links to other dialogues occur whenever a connection between both dialogues is obvious. Pushing the buttons that identify the links opens up the other dialogues.

OPTIONS

The supported options are the standard X Toolkit options as described in *X*(1) section **Options**. Furthermore, *qmon* supports:

-cmap

Installs a private color map for *qmon.* This is sometimes useful if other applications have already allocated lots of colors and if *qmon,* therefore, prints corresponding error messages. **Note**, however, that using a private color map will result in color map switches whenever you enter or leave *qmon* windows.

-fontFamily {big|medium|small}

Notifies *qmon* to use different sized font families for different resolution sizes.

-help

Displays usage information.

-version

Display version information for the *qmon* command.

-nologo

Startup without logo.

DIALOGUES

Job Control

The **Job Control** dialogue provides a folder of tabulated lists of the still pending jobs, already running jobs and recently finished jobs. The dialogue allows for detailed information on the jobs as well as for the deletion and suspension of jobs being selected. In addition the job control dialogue offers links to the **Submit** dialogue in order to submit new jobs or to change attributes of pending jobs (**Qalter** button). The shown displayed fields in the tabular display and the jobs displayed can be customized by pressing the **Customize** button. This customization can be saved to the *~/.qmon_preferences* file and is used on following startups for the initial configuration of the **Job Control** dialogue.

Queue Control

The **Queue Control** dialogue with its sub-dialogue hierarchy enables the user to control the status of the Altair Grid Engine queues being actually configured in the system and allows the administrator to add new queues or to modify or delete already existing ones. The **Queue Control** dialogue shows a tabbed view of Cluster Queues or their corresponding Queue Instances in a tabular way.

By pushing the **Add**, **Clone** or **Modify** button, a sub-dialogue for configuring Altair Grid Engine queues is opened. A queue needs to be selected to use the clone and modify operation. The configuration sub-dialogue allows for definition of the queue and its attributes. The configuration sub-dialogue contains in the upper section the queue name and a list of hosts or hostgroups for which this queue contains the configuration. The lower section contains a list of hosts or hostgroups at the left, where the default attributes are shown when the "@/" entry is highlighted. These default attributes can be overruled on a hostgroup or host basis by enabling attributes and changing their values after this hostgroup or host entry has been added in the "Attributes for Host/Hostgroup" listbox. The queue configuration parameters (see *sge_queue_conf*(5)) are subdivided in different categories (**General Configuration**, **Execution Methods**, **Checkpointing**, **Load/Suspend Thresholds**, **Limits**, **Complexes**, **User Access**, **Project Access, Subordinate Queues, Owners) which are selectable by the tab widget area presented in the lower region of the queue configuration sub-dialogue. The administrator may select previously configured settings from already existing queues (Clone button). By pushing the Ok button, the definitions are registered with** *sge_qmaster*(8). The **Queue Control** dialogue can be customized in a similar way as the **Job Control** dialogue. The settings applied here are also saved in *~/.qmon_preferences*.

Submit

The **Job Submission** dialogue serves for submitting batch and interactive jobs and is also invoked when changing attributes of pending jobs from the **Job Control** dialogue explained above (**Qalter** button). To toggle between batch and interactive jobs please use the **Batch/Interactive** button at the top of the button column on the right side of the **Job Submission** screen.

The dialogue consists of a folder containing two job preparation dialogue pages. The most frequently used parameters in the course of a job submission are offered on the **General** page. A job script has to be defined, all other fields are optional. If the job demands for specification of advanced requirements, the **Advanced** tab can be used to switch to an enhanced parameter display.

If resource requirements are mandatory for the job, the **Request Resources** icon button has to be used to pop up the **Requested Resources** sub-dialogue. This sub-dialogue allows for selection of the required resources of the job and for definition of the quantities in which this resources are to be provided. The **Available Resources** are constituted by those complex attributes being declared *requestable* (see *sge_complex*(5) for details). Resource requirements can be made **Hard**, i.e. they must be met before a job can be started in a queue, or **Soft**, i.e. they are granted on an as available basis.

Closing the **Requested Resources** sub-dialogue with the done button books the specified requirement for the job. Pushing the **Submit** button on the top level **Submit** dialogue submits the job.

Complex Config

The **Complex Config** allows the administrator to add new complex attributes or to modify or delete existing ones (see *sge_complex*(5)). The complex configuration dialogue provides a tabulated list of the complex attribute entries and an input region for the declaration of

new or modified entries. The **Add** button updates the tabulated list with a new attribute. The **Modify** button changes a highlighted existing entry. The **Delete** button removes the highlighted attributes. The **Load and Save** buttons allow saving or reloading the whole attribute list to or from file. The **Ok** button registers the additional or modified complex attributes with *sge_qmaster*(8).

Host Config

Four types of host related lists can be maintained via the **Host Config** dialogue:

Administration Hosts
Submit Hosts
Host Groups
Execution Hosts

The host list to be manipulated is selected via clicking at one of the tabs named correspondingly. The first two host lists only provide for adding or deleting entries, thereby allowing administrative or submit permission for the hosts on the lists, or denying it otherwise respectively. The host group list allows the manipulation of host groups. Host groups can reference either other host groups or hosts. The execution host list entries in addition provide the ability to define scaling factors for the load sensors, consumable complex attributes and access attributes (access, xaccess and projects, xprojects) as described in *sge_complex*(5). CPU, memory and I/O usage reported for running jobs can be scaled in addition and the relative performance of a host can be define with the **Resource Capability Factor** (see *sge_host_conf*(5)).

Cluster Config

This dialogue maintains the cluster global configuration as well as host specific derivatives (see *sge_conf*(5)). When opened, the dialogue displays a selection list for all hosts which have a configuration assigned. The special name “global” refers to the cluster global configuration. By pushing the **Add/Modify** button a sub-dialogue is opened, which allows for modification of the cluster configuration. For host specific configurations the ‘global’ host specific configuration fields are set insensitive and only the allowed parameters can be manipulated.

Scheduler Config

The **Scheduler Configuration** dialogue provides the means to change the behavior of the Altair Grid Engine scheduler thread part of *sge_qmaster*(8) process. The dialogue contains a representation for all scheduler configuration parameters as described in *sge_sched_conf*(5). It is subdivided in the two sections **General Parameters** and **Load Adjustments** which can be selected via the corresponding tabs. The **Ok** button registers any changes with *sge_qmaster*(8).

Calendar Config

The **Calendar Config** allows the administrator to add new calendars or to modify or delete existing ones (see *sge_calendar_conf*(5)). The dialogue offers a selection list for the existing calendars and displays the configuration of the one being selected. By pushing the **Delete** button, the selected calendar is deleted from the configuration. Pushing the **Add**/**Modify** button will open a calendar configuration dialogue, which allows to create new calendars or which provides the means to change the existing ones. The **Ok** button registers the additional or modified calendar with *sge_qmaster*(8).

User Config

User permissions are controlled via the **User Config** dialogue. The tab widget in the left section of the dialogue allows for selecting between

Configuration of **Manager** accounts.
Configuration of **Operator** accounts.
Definition of **Usersets**.
Definition of **User** accounts.

Those user accounts added to the list of manager or operator accounts are given permission to act as managers or operators respectively when accessing Altair Grid Engine under their own account.

The userset lists are used together with the **user_lists** and **xuser_lists** host, queue, project and cluster configuration parameters (see *sge_queue_conf*(5), *sge_project*(5) and *sge_conf*(5)) to control access of users to hosts, queues, projects and the entire cluster. A userset is just a collection of user names and UNIX group names. Group names are identified by prefixing them with a "@" sign. The already defined usersets are displayed in a selection list. These lists can be modified and new lists can be created using the **Userset** definition dialogue.

Usersets can be used as **Access List** and/or as **Department** required for the so called **Functional Policy** and **Override Policy** (see **Ticket Config** below).

User names can be added to the system as entries to the Altair Grid Engine user database (see *sge_user*(5)), This can be done with the **User** sub-dialogue.

The **Tickets** button in the button list on the right side of the dialogue opens the **Ticket Config** dialogue (see below).

PE Config

Parallel environment (PE) interfaces can be configured with this dialogue. PE interfaces are necessary to describe the way how parallel programming environments like PVM (Parallel Virtual Machine), MPI (Message Passing Interface) or shared memory parallel systems are to be instantiated and to impose access restrictions onto the PEs. When the dialogue is opened a list of the already configured PEs is displayed together with the current configuration (see *sge_pe*(5)) of the selected PE interface. To add new PE interfaces or to modify existing ones, an **Add** and a **Modify** button is available which opens a PE interface configuration sub-dialogue. After applying the changes and quitting this sub-dialogue with the **OK** button, the new or modified PE interface is registered with *sge_qmaster*(8).

Checkpoint Config

Checkpointing environment interfaces can be configured with this dialogue. Checkpointing environments are necessary to describe the attributes which the different checkpointing methods and their derivatives on various operating system platforms supported by Altair Grid Engine have. When the dialogue is opened a list of the already configured checkpointing environments is displayed together with the current configuration (see *sge_checkpoint*(5)) of the selected checkpointing environment. To add new checkpointing environment or to modify existing ones, an **Add** and a **Modify** button is available which opens a checkpointing environment configuration sub-dialogue. After applying the changes and quitting this sub-dialogue with the **OK** button, the new or modified checkpointing environment is registered with *sge_qmaster*(8).

Ticket Conf

This dialogue offers an overview and editing screen for allocating tickets to the share-based, functional and override scheduling policies.

The **Deadline Job** button opens the **User Conf** dialogue box. Please change to the Userset sub-dialogue and select the userset named “deadlineusers”. Only users of this userset may submit deadline jobs.

The **Share Tree Policy** button opens the dialogue for creating and editing the Altair Grid Engine share tree (see *sge_share_tree*(5) and *sge_sched_conf*(5) for a description of the configuration parameters).

The **Functional Policy** button opens the dialogue for creating and editing the allocation of the functional shares (see *sge_sched_conf*(5), *sge_access_list*(5), *sge_project*(5), *sge_queue_conf*(5) and *sge_user*(5) for a description of the different types of functional shares and the configurable weighting parameters).

The **Override Policy** button opens the dialogue for creating and editing the allocation of override tickets (see *sge_access_list*(5), *sge_project*(5), *sge_queue_conf*(5) and *sge_user*(5) for a description of the different types of override tickets).

Project Conf

This button opens a dialog for creating projects.
The dialogue offers a selection list for the existing projects and displays the configuration of the one being selected. By pushing the **Delete** button, the selected project is deleted from the configuration. Pushing the **Add**/**Modify** button will open a project configuration dialogue, which allows to create new projects or which provides the means to change the existing ones. Project configuration in essence means giving or denying access to a project for usersets (see **User Conf** above as well as *sge_project*(5)). The Ok button registers the additional or modified project with *sge_qmaster*(8).

Browser

The **Object Browser** dialogue's purpose is manifold: First of all, Altair Grid Engine and *qmon* messages such as notification of error or success concerning a previously taken action can be displayed in the dialogue's output window. Also the standard output and the standard error output of *qmon* can be diverted to the **Object Browser** output window.

Additionally the **Object Browser** can be used to display continuous information about Altair Grid Engine objects as the mouse pointer moves over their representation as icons or table entries in other *qmon* dialogues. Currently, only the display of the configuration of two Altair Grid Engine objects in two separate dialogues is supported:

- Queue configurations are displayed as soon as the mouse pointer enters a queue icon in the top level **Queue Control** dialogue (see above). This facility is activated by pushing the **Queue** button in the **Object Browser** dialogue.
- Detailed job information is printed as soon as the user moves the mouse pointer over a line in the **Job Control** dialogue (see above) being assigned to a running or pending job.
- Additionally job scheduling information is displayed in the browser if the **Why ?** button in the **Job Control** dialogue is pressed. In this case the Browser dialogue is opened implicitly and any scheduling related information is displayed.

Exit
The **Exit** icon button is not linked with a dialogue. Its sole purpose is to close all active *qmon* dialogues and to exit the application.

RESOURCES

The available resources, their meaning and the syntax to be followed in order to modify them are described in the default *qmon* resource file (see the section **Files** below for the location of the resource file).

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qmon* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

XFILESEARCHPATH

If the Qmon configuration file is changed (e.g. in order to alter color settings), it has to be included in this path (export XFILESEARCHPATH=$SGE_ROOT/qmon/Qmon).

RESTRICTIONS

If the line to be entered in an editing window is longer than the width of the window, then the text just runs off the end of the window.

FILES

<sge_root>/qmon/Qmon Qmon sample resources file

/usr/lib/X11/defaults/Qmon

Qmon system resources file

$HOME/Qmon

Qmon user resources file

$HOME/.qmon_preferences

Qmon job/queue customization file

SEE ALSO

sge_intro(1), *sge_conf*(5), *sge_access_list*(5), *sge_pe*(5), *sge_calendar_conf*(5), *sge_complex*(5), *sge_project*(5), *sge_queue_conf*(5), *sge_sched_conf*(5), *sge_user*(5), *sge_qmaster*(8).

COPYRIGHT

See *sge_intro*(1) and the information provided in <sge_root>/3rd_party/qmon for a statement of further rights and permissions and for credits to be given to public domain and freeware widget developers.

qping

NAME

qping - check application status of Altair Grid Engine daemons.

SYNTAX

qping -help -from <string> -to <string> -format <string> -rr <res_host> -noalias [-ssl | -tcp] [[-i <interval> -info -f] | [[-dump_tag <tag> [<param>]] -dump -nonewline]] <host> <port> <name> <id>

DESCRIPTION

Qping is used to validate the runtime status of a Altair Grid Engine service daemon. The current Altair Grid Engine implementation allows one to query the Altair Grid Engine sge_qmaster daemon and any running Altair Grid Engine sge_execd daemon. The qping command is used to send a SIM (Status Information Message) to the destination daemon. The communication layer of the specified daemon will respond with a SIRM (Status Information Response Message) which contains status information about the consulted daemon.

The qping -dump and -dump_tag options allowing an administrator to observe the communication protocol data flow of a Altair Grid Engine service daemon. The qping -dump instruction must be started with root account and on the same host where the observed daemon is running.

OPTIONS

-help

Prints a list of all options.

-version

Display version information for the *qping* command.

-from <string>

This is an output filter useful for the -dump option. It will only show data packages received from communication endpoints containing the string specified with <string>.

-to <string>

This is an output filter useful for the -dump option. It will only show data packages sent to communication endpoints containing the string specified with <string>.

-format <string>

This is an output filter useful for the -dump option. It will only show data packages matching the data format name specified with <string> parameter.

-rr <res_host>

This option is used to verify host name resolving problems. qping will contact the service running on the specified <host> parameter in order to resolve the host name specified with the <res_host> option. On success the exit value zero is returned and the resulting name is printed out. On errors qping returns with 1 and prints out the occurred errors.

-noalias

Ignore host_aliases file, which is located at $SGE_ROOT/$SGE_CELL/common/host_aliases. If this option is used it is not necessary to set the SGE_ROOT environment variable.

-ssl

This option can be used to specify an SSL (Secure Socket Layer) configuration. The qping will use the configuration to connect to services running SSL. If the SGE settings file is not sourced, you have to use the -noalias option to bypass the need for the SGE_ROOT environment variable.

The following environment variables are used to specify your certificates:

environment variable	description
SSL_CA_CERT_FILE	CA certificate file
SSL_CERT_FILE	certificates file
SSL_KEY_FILE	key file
SSL_RAND_FILE	rand file

-tcp

This option is used to select TCP/IP as the protocol used to connect to other services.

-i <interval>

Set qping interval time. The default interval time is one second. Qping will send a SIM (Status Information Message) on each interval time.

-info

Show full status information (see option **-f** for more information) and exit. The exit value 0 indicates no error. On errors qping returns with 1.

-f

Show full status information on each ping interval. The first output line shows the date and time of the request.

The following status info might be printed:

Status	Description
SIRM version	Internal version number of the SIRM (Status Information Response Message)
SIRM message id	Current message id for this connection
start time	Start time of daemon. The format is MM/DD/YYYY HH:MM:SS (nr of seconds since 01.01.1970)
run time [s]	Run time in seconds since start time
messages in read buffer	Nr. of buffered messages in communication buffer. The messages are buffered for the application (daemon). When this number grows too large the daemon is not able to handle all messages sent to it.
messages in write buffer	Nr. of buffered messages in the communication write buffer. The messages are sent from the application (daemon) to the connected clients, but the communication layer was not able to send the messages yet. If this number grows too large, the communication layer is not able to send them as fast as the application (daemon) wants the messages to be sent.
nr. of connected clients	This is the number of actual connected clients to this daemon. This also implies the current qping connection.
status	The status value of the daemon. This value depends on the application which reply to the qping request. If the application does not provide any information the status is 99999.

Status	Description
info	Status message of the daemon. This value depends on the application which reply to the qping request. If the application does not provide any information the info message is “not available”.
malloc	This line shows malloc memory statistics if available.
jemalloc	On platforms where we use jemalloc instead of the OS malloc library this line will be printed instead of the malloc line. It shows jemalloc specific statistics.
Monitor	If available, displays statistics on a thread. The data for each thread is displayed in one line. The format of this line can be changed at any time. Only the master implements the monitoring.
cached ns lookups	In a default installation the communication library is caching host resolving results. If available this line displays the number of resolved hostnames in the cache.
nr of work threads	The current number of work threads the communication library is using. It shows also the min and max values if a range is configured for the thread pool size. See **wp_threads** paramter in *sge_bootstrap*(5) man page or **cl_wp_threads** for execd_params and qmaster_params in *sge_conf*(5) man page.

sge_qmaster daemon specific status information values:

sge_qmaster status	Value	Description
status	0	No unusual timing situation.
status	1	One or more threads has reached warning timeout. This may happen when at least one thread does not increment his time stamp for a not usual long time. A possible reason for this is a high workload for this thread.
status	2	One or more threads has reached error timeout. This may happen when at least one thread has not incremented his time stamp for longer than 10 minutes.
status	3	The time measurement is not initialized.
info	-	The info message contains information about the qmaster threads followed by a thread state and time information. Each time when one of the known threads pass through their main loop the time information is updated. Each thread might show a specific thread error state (R=Running, W=Warning and E=Error).
Monitor	-	Show the monitoring info if **MONITOR_TIME** parameter is enabled. See also *sge_conf*(5) man page.

sge_execd daemon specific status information values:

sge_execd status	Value	Description
status	0	No unusual timing situation.
status	1	Dispatcher has reached warning timeout. This may happen when the dispatcher does not increment his time stamp for a unusual long time. A possible reason for this is a high workload.
status	2	Dispatcher has reached error timeout. This may happen when the dispatcher has not incremented his time stamp for longer than 10 minutes.
status	3	The time measurement is not initialized.
info	-	The info message contains information for the execd job dispatcher. Each time when the dispatcher pass through their its loop the time information is updated. A specific error state (R=Running, W=Warning and E=Error) might be shown.

-dump_tag <tag> [param]

This option has the same the same meaning as -dump, but can provide more information by specifying the debug level and message types qping should print:

Option	Description
-dump_tag ALL <debug level>	This option shows all possible debug messages (APP+MSG) for the debug levels, ERROR, WARNING, INFO, DEBUG and DPRINTF. The contacted service must support this kind of debugging. This option is not currently implemented.
-dump_tag APP <debug level>	This option shows only application debug messages for the debug levels, ERROR, WARNING, INFO, DEBUG and DPRINTF. The contacted service must support this kind of debugging. This option is not currently implemented.
-dump_tag MSG	This option has the same behavior as the -dump option.

-dump

This option allows an administrator to observe the communication protocol data flow of a Altair Grid Engine service daemon. The qping -dump instruction must be started as root user and on the same host where the observed daemon is running.

The output is written to stdout. The environment variable **SGE_QPING_OUTPUT_FORMAT** can be set to hide columns, set a default column width or to set a hostname output format. The value of the environment variable can be set to any combination of the following specifiers separated by a space character:

Value	Description
“h:X”	hide column X
“s:X”	show column X
“w:X:Y”	width of column X to Y
“hn:X”	set hostname output parameter X (Allowed X values are “long” or “short”)

Start qping -help to see which columns are available and currently enabled.

With this environment variable it is also possible to enable a full “message content dump” which might be helpful when debugging problems.

Each printed line shows message specific information for a single data package send or received by the observed daemon. Per default the following output columns are shown:

Column name	Description
time	time when message was received at qping client
local	name of the local component (endpoint)
d.	direction “->”(outgoing) or “<-”(incoming)
remote	name of the remote component (endpoint)
format	message format string
ack type	message acknowledge type
msg tag	message tag string
msg id	message id
msg rid	message response id
msg len	message length
msg time	time when message was created or received
msg ltime	time message lingered in commlib
con count	number of connected clients
msg ulen	length of uncompressed message
msg cratio	compressed/uncompressed message ratio
msg mod	message mode (none=default, zlib=zlib compressed)

-nonewline

Dump output will not have a linebreak within a message and binary messages are not unpacked.

<host>

Host where daemon is running.

<port>

Port which daemon has bound (used sge_qmaster/sge_execd port number).

<name>

Name of communication endpoint (“qmaster” or “execd”). A communication endpoint is a triplet of “hostname/endpoint name/endpoint id” (e.g. hostA/qmaster/1 or sub-host/qstat/4).

<id>

Id of communication endpoint (“1” for daemons).

EXAMPLES

```
> qping walnut-vb $SGE_EXECD_PORT execd 1
06/07/2018 21:54:01 endpoint walnut-vb/execd/1 at port 11301 is up since 24632 seconds
06/07/2018 21:54:02 endpoint walnut-vb/execd/1 at port 11301 is up since 24633 seconds
06/07/2018 21:54:03 endpoint walnut-vb/execd/1 at port 11301 is up since 24634 seconds
06/07/2018 21:54:04 endpoint walnut-vb/execd/1 at port 11301 is up since 24635 seconds

>qping -info walnut-vb $SGE_QMASTER_PORT qmaster 1
06/07/2018 22:02:37:
SIRM version:             0.2
SIRM message id:          1
start time:               06/07/2018 15:03:23 (1528376603)
run time [s]:             25154
messages in read buffer:  0
messages in write buffer: 0
nr. of connected clients: 7
status:                   1
info:                     MAIN: E (25153.44) | signal000: E (25152.61) ...
malloc:                   arena  (0) |ordblks  (1) | smblks  (0) | hblksr  (0) ...
Monitor:                  disabled
cached ns lookups:        10
nr of work threads:       3 (min=3, max=7)


> qping -info walnut-vb $SGE_EXECD_PORT execd 1
06/07/2018 21:59:27:
SIRM version:             0.2
SIRM message id:          1
start time:               06/07/2018 15:03:29 (1528376609)
run time [s]:             24958
messages in read buffer:  0
messages in write buffer: 0
nr. of connected clients: 2
status:                   1
info:                     MAIN: E (24957.64) | WARNING
malloc:                   arena  (0) |ordblks  (1) | smblks  (0) | hblksr  (0) ...
```

```
Monitor:                  disabled
cached ns lookups:        1
nr of work threads:       0 (min=0, max=0)

# SGE_QPING_OUTPUT_FORMAT="h:2 h:6 h:7 h:9 h:11 h:14 w:4:10:18" ; export SGE_QPING_OUTPUT_FORMA
# qping -dump -format bin -from qstat -to qstat walnut-vb $SGE_QMASTER_PORT qmaster 1
open connection to "walnut-vb/qmaster/1" ... no error happened
           time|d.|remote          |format|msg id|msg len|con count|msg ulen|msg cratio|msg mod
---------------|--|----------------|------|------|-------|---------|--------|----------|-------
13:47:14.532523|<-|nut-vb/qstat/524|   bin|     1|    546|        6|    3853|     7.057|   zlib
13:47:14.545009|->|nut-vb/qstat/524|   bin|     2|   8630|        6|  250725|    29.053|   zlib
13:47:18.992911|<-|nut-vb/qstat/525|   bin|     1|    501|        6|    2065|     4.122|   zlib
13:47:19.094567|->|nut-vb/qstat/525|   bin|     2|  34676|        6| 3075638|    88.696|   zlib

#SGE_QPING_OUTPUT_FORMAT="h:2 h:6 h:7 h:9 h:11 h:14 s:12 w:4:10:18" ; export SGE_QPING_OUTPUT_F
# qping -dump -from "mango-vb/execd" -to "none" walnut-vb $SGE_QMASTER_PORT qmaster 1
open connection to "walnut-vb/qmaster/1" ... no error happened
           time|d.|remote          |format|msg id|msg len|msg dump            |con count|msg ul
---------------|--|----------------|------|------|-------|--------------------|---------|------
14:17:50.883086|<-|mango-vb/execd/1|   bin|   198|    288|binary message data |        5|     2
--- BINARY block start ------------------------------------------------------------------------
   0000000010020000 100035f000000001 000000027265706f 7274206c69737400    ........ ..5..... ...
   0000000100000000 00001a9000000002 000000016d616e67 6f2d766200100035    ........ ........ ...
...
      unpacked report request (binary buffer length 288):

List: <report list>   #Elements: 2
-------------------------------
REP_type                                (Ulong)     = 1
REP_host                                (Host)      = mango-vb
REP_list                                (List)      = full {

   List: <element>   #Elements: 1
   ------------------------------
   LR_name                                 (String)    = cpu
   LR_value                                (String)    = 0.400000
   LR_global                               (Ulong)     = 0
   LR_static                               (Ulong)     = 0
   LR_host                                 (Host)      = mango-vb
}
REP_version                             (Ulong)     = 268449264
REP_seqno                               (Ulong64)   = 177
REP_time                                (Ulong64)   = 1528892270881
-------------------------------
REP_type                                (Ulong)     = 3
REP_host                                (Host)      = mango-vb
REP_list                                (List)      = full {

   List: <execd config list copy>   #Elements: 2
```

```
    ------------------------------
    CONF_name                          (Host)      = global
    CONF_version                       (Ulong)     = 5
    CONF_entries                       (List)      = empty
    CONF_ctx                           (List)      = empty
    ------------------------------
    CONF_name                          (Host)      = mango-vb
    CONF_version                       (Ulong)     = 1
    CONF_entries                       (List)      = empty
    CONF_ctx                           (List)      = empty
}
REP_version                        (Ulong)     = 268449264
REP_seqno                          (Ulong64)   = 177
REP_time                           (Ulong64)   = 1528892270881
```

ENVIRONMENTAL VARIABLES

SGE_ROOT Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL If set, specifies the default Altair Grid Engine cell.

SGE_QPING_OUTPUT_FORMAT Used for option **-dump** column formatting

SEE ALSO

sge_intro(1), *sge_host_aliases*(5), *sge_execd*(8), *sge_qmaster*(8)

COPYRIGHT

qquota

NAME

qquota - shows current usage of Altair Grid Engine resource quotas

SYNTAX

qquota [**-h wc_host|wc_hostgroup,...**] [**-help**] [**-l resource_name,...**] [**-u wc_user,...**] [**-P wc_project,...**] [**-pe wc_pe_name,...**] [**-q wc_cqueue,...**] [**-si session_id**] [**-xml**]

DESCRIPTION

qquota shows the current Altair Grid Engine resource quota sets. Only resource quota sets with a positive usage count or a static limit are printed.

Selection options allow you to filter for specific hosts, cluster queues, projects, parallel environments (pe), resources or users. Without any option *qquota* will display a list of all resource quota sets for the calling user.

OPTIONS

-h wc_host|wc_hostgroup,...

Display only resource quota sets that matches with the hosts in the given wildcard host or hostgroup list. Find the definition of **wc_host** and **wc_hostgroup** in *sge_types*(1).

-jc jc_list

Display only resource quota sets that matches with the job class list. Find the definition of **jc_list** in *sge_types*(1).

-help

Prints a listing of all options.

-version

Display version information for the *qquota* command.

-l resource_name,...

Display only resource quota sets being matched with the resources in the given resource list.

-u wc_user,...

Display only resource quota sets being matched with the users in the given wildcard user list. Find the definition of **wc_user** in *sge_types*(1).

-P wc_project,...

Display only resource quota sets being matched with the projects in the given wildcard project list. Find the definition of **wc_project** in *sge_types*(1).

-pe wc_pe_name,...

Display only resource quota sets being matched with the parallel environments (pe) in the given wildcard pe list. Find the definition of **wc_pe_name** in *sge_types*(1).

-q wc_cqueue,...

Display only resource quota sets being matched with the queues in the given wildcard cluster queue list. Find the definition of **wc_cqueue** in *sge_types*(1).

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-xml

This option can be used with all other options and changes the output to XML. The schema used is referenced in the XML output. The output is printed to stdout.

OUTPUT FORMATS

A line is printed for every resource quota with a positive usage count or a static resource. The line consists of

*the resource quota - rule set name/rule name or number of rule in ruleset

*the limit - resource name, the number of used and available entities of that resource

*the effective resource quota set filter

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qquota* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which sge_qmaster(8) is expected to listen for communication requests. Most installations will use a services map entry for the service “sge_qmaster” instead to define that port.

FILES

- *<SGE_ROOT>/<SGE_CELL>/common/act_qmaster* - Altair Grid Engine master host file
- *<SGE_ROOT>/<SGE_CELL>/common/sge_qquota* - cluster qquota default options
- *$HOME/.sge_qquota* - user qquota default option

SEE ALSO

sge_intro(1), *qalter*(1), *qconf*(1), *qhold*(1), *qmod*(1), *qstat*(1), *qsub*(1), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_execd*(8), *sge_qmaster*(8), *sge_shepherd*(8).

COPYRIGHT

qralter

NAME

qrsub -

Submit an Advance Reservation (AR) to Altair Grid Engine.

qralter -

Modify an Advance Reservation (AR) of Altair Grid Engine.

SYNTAX

qrsub options

qralter options ar_id|ar_name

DESCRIPTION

Qrsub provides a means for operators, managers, or users listed in the ACL (see *sge_access_list*(5)) "**arusers**" to create an Advance Reservation (AR) or a Standing Reservation (SR) in the Altair Grid Engine queuing system. ARs allow reservation of particular consumable resources for future use. These reserved resources are only available to jobs requesting the AR, and the scheduler ensures the availability of the resources when the start time is reached. Jobs requesting the AR can only use the reserved consumable resources. SRs are recurring ARs which follow a given calendar. All ARs within one SR have the same ID.

During AR submit time the Altair Grid Engine queuing system selects the best-suited queues for the AR request, and then reserves the desired amount of resources. For a reservation, all queues that are not in an orphaned state are considered to be suited. The AR will be granted only if the AR request can be fulfilled.

ARs will be deleted either automatically when the end time is reached, or manually using *qrdel*. In both cases, first all jobs requesting the AR will be removed, and then the AR itself. Already-granted ARs can be shown with *qrstat*.

Note: To make AR behavior predictable, it is necessary to have reserved resources available at the time of AR start. This is done by keeping jobs with an unlimited runtime limit separated from ARs, and not considering resources used by such jobs for reservation.

Note: Resource Quotas are not considered for AR queue selection or for jobs requesting an AR.

When an AR is successfully added to the Altair Grid Engine queuing system, *qrsub* returns a unique integer ID referring the the newly-created AR. The highest AR ID is 9999999. If the

highest ID is reached, a wraparound happens and the next unused ID, starting with 1, will be used.

For *qrsub*, the administrator and the user may define default request files (analogous to Altair Grid Engine_request for qsub), which can contain any of the possible command-line options.
A cluster-wide default request file is optional. If such a default request file is used, it must be placed under
$SGE_ROOT/$SGE_CELL/common/sge_ar_request (global defaults file).
A user-private default request file is optional. If it is used, it must be placed under
$HOME/.sge_ar_request (user private defaults file).

qralter can be used to change the attributes of an advance reservation. Whether an attribute can be changed depends on the state of the AR:

All attributes can be changed for ARs that are pending or running, but there must be no jobs running in the AR. It might become necessary to reschedule the AR, e.g. if resource requests (-l / -masterl) or queue requests (-q / -masterq) are modified. If rescheduling is not possible because the requested resources are not available in the given time frame, qralter will print an error message and the AR will not be modified.

If an AR is already running and has running jobs:

- simple modifications not affecting the reserved resources, such as modifying the name (-N) or the account string (-A), will always work
- modifying start time (-a), end time (-e), or duration (-d) will work, if the resources held by the AR will also be available in the new time frame. Reducing the time frame will always be accepted.
- if rescheduling of the AR would be necessary because e.g. resource requests will be modified (-l / -masterl) or the given set of resources will not be available in an extended time frame (-e / -d), qralter will print an error message and the AR will not be modified.

With the qmaster_param AR_DETACH_JOBS_ON_RESCHEDULE it is possible to have running jobs being detached from the AR and thus allowing re-scheduling; see sge_conf(5).

OPTIONS

-a date_time

Defines the activation (start) date and time of an AR. The option is not required. If it is omitted, the current date_time is assumed. Either a duration or end date_time must be specified for ARs. For details about date_time please see *sge_types*(1) For SRs the start time is optional and determines the earliest possible allocation time of an AR (but not the actual beginning of the first AR).

-A account_string

Identifies the account to which the resource reservation of the AR should be charged. For "**account_string**" value details please see the "**name**" definition in *sge_types*(1). In the ab-

sence of this parameter Altair Grid Engine will place the default account string “sge” in the accounting record of the AR.

-cal_week weekly_calendar

If a calendar is defined it is a Standing Reservation request. The weekly calendar has the same format as the calendar object (see *sge_calendar_conf*(5)) with the exception that only the “**on**” state is allowed. The weekly calendar defines the time ranges for which AR instances should be created. The scheduler tries to allocate a fixed amount of ARs. That amount can be influenced with the depth (“**-cal_depth**”) parameter. All ARs within an SR have the same ID. Jobs running within one AR are deleted at AR end (as with ARs) but queued jobs stay queued waiting for the next AR occurrence within the SR. Giving a start time, end time, or duration is optional for SRs. If given, the start time determines the earliest allowed start time for the first AR. The end time determines the last possible end time of the last AR within the SR. The duration specifies the end time. If one AR within the SR ends, new ARs are scheduled automatically, so that the total amount of allocated ARs are equal to the SR depth.

-cal_depth number

Specifies the depth of the SR. The depth determines how many ARs are allocated (scheduled) at any point in time in the future. ARs which cannot get enough resources are unallocated and not counted. If one AR ends, further ARs are allocated so that the depth is restored. If an AR cannot be allocated it will go into an Error state. Default for “**-cal_depth**” is 8.

-cal_jmp number

The jmp parameter determines how many of the first calendar entries can be jumped over if they cannot be allocated for ARs due to missing resources at SR submission time. If the first n ARs of a standing reservation cannot be allocated and “**-cal_jmp**” is less than n, the submission of the SR will be rejected. Default for “**-cal_jmp**” is 0.

-ckpt ckpt_name

Selects the checkpointing environment (see *sge_checkpoint*(5)) the AR jobs may request. Using this option guarantees that only queues providing this checkpoint environment will be reserved.

-d time

Defines the duration of the AR. The use of “**-d time**” is optional if “**-e date_time**” is requested. For details about “**time**” definition please see *sge_types*(1). For SRs the duration is optional and determines implicitly the end time, i.e. the last possible end time for the last AR within the SR.

-e date_time

Defines the end date and time of an AR. The use of "**-e date_time**" is optional if "**-d time**" is requested. For details about "**date_time**" definition please see *sge_types*(1). For SRs the end time is optional and determines the last possible end time for the last AR within the SR.

-fr y[es]|n[o]

Specifies the behavior of the AR at AR start time. With ("**-fr yes**") jobs still blocking resources being granted to the AR will be killed. With default behavior ("**-fr no**") jobs holding resources being granted to the AR can continue running and may prevent start of jobs in the AR. In the current implementation only resources requested by the AR via exclusive complexes are freed. The feature is suited for freeing whole queue instances or hosts or even the whole cluster, e.g. for maintenance purposes. Requesting "**-fr yes**" requires operator privileges.

-he y[es]|n[o]

Specifies the behavior when the AR goes into an error state. The AR goes into an error state when a reserved host goes into an unknown state, a queue error happens, or when a queue is disabled or suspended.

A hard error, "**-he yes**", means as long as the AR is in an error state no jobs using the AR will be scheduled. If a soft error, "**-he no**", is specified the AR remains usable with the remaining resources.
By default soft error handling is used.

-help

Prints a list of all options.

-version

Display version information for the command

-jc IGNORE_JC | NO_JC | ANY_JC |

If specified allows filtering of queue instances that may be considered for AR reservation.
By default, if the **-jc** switch is omitted or when **-jc IGNORE_JC** is specified, queues that accept JC **and** non-JC jobs at the same time will be considered for the AR. If the only queues that should be selected are those that do not allow execution of JC jobs, **-jc NO_JC** can be specified.
If the only queues that should be selected are those that allow execution of JC jobs, **-jc ANY_JC** is the right choice.
Additionally it is possible to filter queue instances that accept jobs derived from specific job classes. To achieve this a pattern for job class variants, or a specific job class variant name,

can be specified. This will select only those queue instances where jobs that are derived from a job class variant that matches the pattern are also allowed to be executed.

-l resource=value,...

Creates an AR in a Altair Grid Engine queue; the AR provides the given resource request list. *sge_complex*(5) describes how a list of available resources and their associated valid value specifiers can be obtained.
There may be multiple **-l** switches in a single command.

With AR submission it is possible to overwrite the consumable attribute of a resource description (the complex variable). Overwriting a consumable is only possible for complex variables which are already consumable (one of YES, JOB, HOST).

Syntax: -l resource[=value[{consumable}]][,resource[=value[{consumable}]],...] where consumable is one of YES, JOB, HOST.

Can be combined with **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

-masterl resource=value,...

Available only in combination with parallel jobs, i.e. together with the -pe option.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

qralter does allow changing the value of this option for the reservation; however the modification will only be effective if the job is not running in the reservation.

-masterq wc_queue_list

Available only in combination with parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may

make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qralter does allow changing the value of this option for the reservation; however the modification will only be effective if the job is not running in the reservation.

-m b|e|a|n

Defines or redefines under which circumstances mail is to be sent to the AR owner or to the users listed with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the AR
'e' Mail is sent at the end of the AR
'a' Mail is sent when the AR goes into an error state
'n' No mail is sent. This is the default for *qrsub*

-M user[@host],...

Defines or redefines the list of users to which the qmaster sends mail.

-masterq wc_queue_list

Only meaningful for a parallel AR request when used together with the -pe option.

This option is used to reserve the proper queues to match this request as if it were requested via qsub. A more detailed description of *wc_queue_list* can be found in *sge_types*(1).

-now y[es]|n[o]

This options impacts the queues selection for reservation.
With the "**-now y**" option, only queues with the qtype "INTERACTIVE" assigned will be considered for reservation. "**-now n**" is the default for *qrsub*.

-N name

The name of the AR. The name, if requested, must conform to "**name**" as defined in *sge_types*(1). Invalid names will be denied at submit time.

Note that names have to be unique when the qmaster param **SUBMIT_JOBS_WITH_AR_NAME** is set to true. qralter cannot change the AR name when using ar_name as an argument.

-P project_name

This option binds the AR to a project. Only members of this project are allowed to use this AR.

-w e|v

Specifies the validation level applied to the AR request.

The specifiers e and v define the following validation modes:

'v' Verify: does not submit the AR but prints an extensive validation report

'e' Error: rejects request if requirements cannot be fulfilled. This is the default for *qrsub*

-par allocation_rule

This option can be used with a parallel programming environment (PE).

It can be used to overwrite the allocation rule of the parallel environment into which an AR gets submitted with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel advance reservation.

This option can also be used after a **-petask** switch to overwrite the allocation rule for a specific set of tasks only. If there are multiple task sets defined at the command line, the scheduler will combine all groups with identical allocation rules. The tasks of such a group will be scheduled together. The scheduler will start with the group containing the task with the smallest ID, then will continue with the next group containing the smallest unscheduled task.

Valid allocation rules are **$pe_slots**, **$fill_up**, **$round_robin**, and a positive number as a **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

-pe parallel_env n[-[m]]|[-]m,...

Parallel programming environment (PE) to select for the AR queue reservation. Please see the details of a PE in *sge_pe*(5).

-petask tid_range_list ...

Available for *qrsub* and *qralter*.

Can be used to submit parallel advance reservations (submitted with **-pe** request); indicates that all resource requirements specified via **-q** and **-l**, and in combination with **-hard**, **-soft**, and **-par** that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the master task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request, the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The master task (ID 0) has a memory request of 4M. All slave tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

qralter can be used in combination with the **-petask** switch to completely replace the previously specified requests for an existing task ID range, as long as no additional restrictions

apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page. Please note also that attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission will be rejected.

-q wc_queue_list

Defines or redefines a list of cluster queues, queue domains, or queue instances that may be reserved by the AR. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-petask** to specify specific queue requirements for individual PE tasks or a group of PE tasks. Find more information in the corresponding sections of this man page.

-rao y[es]|n[o]

Specifies which resources will be considered for the AR at submission/reservation time. When using **-rao yes**, only currently available resources will be reserved. This means using only queues which are not in disabled, suspended, error or unknown state. The default setting for **-rao** is "no". A cluster-wide setting for only reserving available resources can be done via qmaster_param AR_RESERVE_AVAILABLE_ONLY; see *sge_conf*(5).

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-u [username|@access_list],...

Specifies the users allowed to submit jobs requesting the AR. The access is specified by a comma-separated list containing UNIX users or ACLs (see *sge_access_list*(5)). An ACL name is prefixed with an '@' sign.
By default only the AR owner is allowed to submit jobs requesting the AR.

Note: Only queues where all users specified in the list have access are considered for reservation (see *sge_queue_conf*(5)).

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell, *qsub*, *qsh*, *qlogin*, or *qalter* use (in order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the TCP port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead of defining the port.

FILES

$SGE_ROOT/$SGE_CELL/common/sge_ar_request

global defaults file

$HOME/.sge_ar_request

user-private defaults file

SEE ALSO

qrdel(1), *qrstat*(1), *qsub*(1), *sge_types*(1), *sge_checkpoint*(5), *sge_complex*(5), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_pe*(5), *sge_resource_quota*(5), *sge_calendar_conf*(5).

COPYRIGHT

qrdel

NAME

qrdel - delete Altair Grid EngineAdvance Reservations (AR)

SYNTAX

qrdel [**-f**] [**-help**] [**-si session_id**] [**-u wc_user_list**] **wc_ar_list**

DESCRIPTION

Qrdel provides a means for a user/operator/manager to delete one or more Advance Reservations (AR) or Standing Reservations (SR) A manager/operator can delete ARs belonging to any user, while a regular user can only delete his or her own ARs. If a manager wants to delete another user's AR, the manager can specify the AR id. By default, "qrdel wc_ar_name" will delete only the ARs belonging to that user. A manager is able to delete another user's AR via "-u wc_user_list". Jobs referring to an AR tagged for deletion will also be removed. Only if all jobs referring to an AR are removed from the Altair Grid Enginedatabase will the AR also be removed.

Qrdel deletes ARs in the order in which the AR identifiers are presented. Find additional information concerning *wc_user_list* and *wc_ar_list* in *sge_types*(1).

OPTIONS

-f

Force action for AR. The AR and the jobs using the AR are deleted from the Altair Grid Enginequeuing system even if the *sge_execd*(8) controlling the AR jobs does not respond to the delete request sent by the *sge_qmaster*(8).

Users which have neither Altair Grid Enginemanager nor operator status can only use the **-f** option for their own ARs.

-help

Prints a list of all options.

-version

Display version information for the *qrdel* command.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-u wc_user_list

Deletes only those ARs which were submitted by users specified in the list of usernames. For managers, it is possible to use **qrdel -u "*"** to delete all ARs for all users. If a manager wants to delete a specific AR for a user, he has to specify the user and the AR id. If no AR is specified, all ARs belonging to that user are deleted.

wc_ar_list

A list of AR id's that should be deleted

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Enginestandard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Enginecell. To address a Altair Grid Engine cell *qrdel* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which the *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a service map entry for the service “sge_qmaster” instead of defining the port.

FILES

- *<SGE_ROOT>/<SGE_CELL>/common/act_qmaster* - Altair Grid Enginemaster host file

SEE ALSO

sge_intro(1), *qrstat*(1), *qrsub*(1), *qsub*(1), *sge_session_conf*(5), *sge_qmaster*(8), *sge_execd*(8).

COPYRIGHT

qresub

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request "m_mem_free" is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host ("m_mem_free_n<node>").

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule "$pe_slots". Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the "m_mem_free" host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule "$pe_slots"). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can't fulfill the "m_mem_free" request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to "one core on the socket" and decrements the amount of memory on this socket ("m_mem_free_n<nodenumber>"). The global host memory "m_mem_free" on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from "m_mem_free" as well as from "m_mem_free_n<socketnumber>", then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don't offer "m_mem_free" memory are skipped (of course hosts that don't offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory ("m_mem_free_n<N>") and enough free cores, binds the job to these cores and sets memory allocation mode to "cores:strict" (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed ("m_mem_total_n<N>"). Binds the job to the complete socket, decrements the memory on that socket at "m_mem_free_n<N>" (as well as host globally on "m_mem_free") and sets the memory allocation strategy to "cores" (preferred usage of socket local memory).

If the parameters request the "m_mem_free" complex, the resulting NUMA node memory requests can be seen in the "implicit_requests" row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

‘n’ none (default)
Switches off validation
‘e’ error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

qrls

NAME

qrls - release Altair Grid Enginejobs from previous hold states

SYNTAX

qrls [**-h** {**u**|**o**|**s**},...] [**-help**] [**job/task_id_list**]

qrls [**-h** {**u((|**o**|**s******},...] [**-help**] **-u user_list**

DESCRIPTION

Qrls provides a means for a user/operator/manager to release so called *holds* from one or more jobs pending to be scheduled for execution. As long as any type of hold is assigned to a job, the job is not eligible for scheduling.

Holds can be assigned to jobs with the *qhold*(1), *qsub*(1) or the *qalter*(1) command.

There are three different types of holds:

user User holds can be assigned and removed by managers, operators and the owner of the jobs.

operator Operator holds can be assigned and removed by managers and operators.

system System holds can be assigned and removed by managers only.

If no hold type is specified with the *-h* option (see below) the user hold is assumed by default.

An alternate way to release holds is the *qalter*(1) command (see the *-h* option).

OPTIONS

-h {u|o|s},...

Releases a u(ser), o(perator) or s(system) hold or a combination there of from one or more jobs.

-help

Prints a listing of all options.

-version

Display version information for the *qrls* command.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session. Find more information concerning sessions in *sge_session_conf*(5).

-u username,...

Modifies the hold state of those jobs which were submitted by users specified in the list of usernames. Managers are allowed to use the **qrls -u "*"** command to modify the hold state for jobs of all users.

If a user uses the **-u** switch, the user may specify an additional **job/task_id_list**.

job/task_id_list

Specified by the following form:

job_id[.task_range][,job_id[.task_range],...]

If present, the *task_range* restricts the effect of the operation to the array job task range specified as suffix to the job id (see the **-t** option to *qsub*(1) for further details on array jobs).

The task range specifier has the form n[-m[:s]]. The range may be a single number, a simple range of the form n-m or a range with a step size.

Instead of *job/task_id_list* it is possible to use the keyword 'all' to modify all jobs of the current user.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qrls* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SEE ALSO

sge_intro(1), *qalter*(1), *qhold*(1), *qsub*(1).

COPYRIGHT

qrsh

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request "m_mem_free" is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host ("m_mem_free_n<node>").

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule "$pe_slots". Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the "m_mem_free" host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule "$pe_slots"). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can't fulfill the "m_mem_free" request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to "one core on the socket" and decrements the amount of memory on this socket ("m_mem_free_n<nodenumber>"). The global host memory "m_mem_free" on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from "m_mem_free" as well as from "m_mem_free_n<socketnumber>", then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don't offer "m_mem_free" memory are skipped (of course hosts that don't offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory ("m_mem_free_n<N>") and enough free cores, binds the job to these cores and sets memory allocation mode to "cores:strict" (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed ("m_mem_total_n<N>"). Binds the job to the complete socket, decrements the memory on that socket at "m_mem_free_n<N>" (as well as host globally on "m_mem_free") and sets the memory allocation strategy to "cores" (preferred usage of socket local memory).

If the parameters request the "m_mem_free" complex, the resulting NUMA node memory requests can be seen in the "implicit_requests" row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

‘n’ none (default)
Switches off validation
‘e’ error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

qrstat

NAME

qrstat - show the status of Altair Grid Engine Advance Reservations (AR)

SYNTAX

qrstat [**-ar ar_list**] [**-help**] [**-si session_id**] [**-u user,...**] [**-explain**]

DESCRIPTION

qrstat shows the current status of the available Altair Grid Engine ARs or Standing Reservations (SRs). The selection option **-ar** allows you to get information about specific AR or SR.

The administrator and the user may define files which can contain any of the options described below. A cluster-wide sge_qrstat file may be placed under "SGE_ROOT/SGE_CELL/common/sge_qrstat". The user private file is searched at the location "$HOME/.sge_qrstat". The home directory request file has the highest precedence over the cluster global file. Command line can be used to override the flags contained in the files.

OPTIONS

-ar ar_list

Prints various information about the ARs identified by given ar_id's or ar_name's in the ar_list.

-explain

Displays the reason for the error state of an AR. Possible reasons are the unknown state of a host or queue instance.

The output format for the alarm reasons is one line per reason.

-help

Prints a listing of all options.

-version

Display version information for the *qrstat* command.

-U [user,...]

Displays all ARs accessible by the users specified. If no usernames were given, all ARs are shown which the current user is allowed to access.

-u user,...

Display information only for those ARs created by the users from the given user list.

The string **$user** is a placeholder for the current username. An asterisk "*" can be used as username wildcard to request that all users' ARs be displayed. The default value for this switch is "**-u $user**".

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-xml

This option can be used with all other options and changes the output to XML. The used schemas are referenced in the XML output. The output is printed to stdout.

-json

This option can be used with all other options and changes the output to JSON. The used schemas are referenced in the JSON output. The output is printed to stdout.

OUTPUT FORMATS

Depending on the presence or absence of the **-ar** option there are two different output formats.

Advance Reservation Summary (without -ar)

Following the header line, a section for each AR or SR is provided. The columns contain information for

- the AR id.
- the name of the AR / SR.
- the current state of the AR (first AR in case of an SR). One of following states 'wWrEd".

w - waiting without error
W - warning (effective - waiting with error)
r - running
E - error (effective - running with error)
d - deleted

- the start time of the AR / SR.
- the end time of the AR / SR.
- the duration of the AR / SR.
- if the given element is a SR or AR.

Detailed Format (with -ar)

The output contains two columns. The first one contains all AR and SR attributes. The second one the corresponding value.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qrstat* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

FILES

SGE_ROOT/SGE_CELL/common/act_qmaster Altair Grid Engine master host file *SGE_ROOT/SGE_CELL/common/sge_q*
cluster qrstat default options *$HOME/.sge_qrstat* user qrstat default options

SEE ALSO

sge_intro(1), *sge_session_conf*(5), *qrsub*(1), *qrdel*(1), *qsub*(1)

COPYRIGHT

qrsub

NAME

qrsub -

Submit an Advance Reservation (AR) to Altair Grid Engine.

qralter -

Modify an Advance Reservation (AR) of Altair Grid Engine.

SYNTAX

qrsub options

qralter options ar_id|ar_name

DESCRIPTION

Qrsub provides a means for operators, managers, or users listed in the ACL (see *sge_access_list*(5)) "**arusers**" to create an Advance Reservation (AR) or a Standing Reservation (SR) in the Altair Grid Engine queuing system. ARs allow reservation of particular consumable resources for future use. These reserved resources are only available to jobs requesting the AR, and the scheduler ensures the availability of the resources when the start time is reached. Jobs requesting the AR can only use the reserved consumable resources. SRs are recurring ARs which follow a given calendar. All ARs within one SR have the same ID.

During AR submit time the Altair Grid Engine queuing system selects the best-suited queues for the AR request, and then reserves the desired amount of resources. For a reservation, all queues that are not in an orphaned state are considered to be suited. The AR will be granted only if the AR request can be fulfilled.

ARs will be deleted either automatically when the end time is reached, or manually using *qrdel*. In both cases, first all jobs requesting the AR will be removed, and then the AR itself. Already-granted ARs can be shown with *qrstat*.

Note: To make AR behavior predictable, it is necessary to have reserved resources available at the time of AR start. This is done by keeping jobs with an unlimited runtime limit separated from ARs, and not considering resources used by such jobs for reservation.

Note: Resource Quotas are not considered for AR queue selection or for jobs requesting an AR.

When an AR is successfully added to the Altair Grid Engine queuing system, *qrsub* returns a unique integer ID referring the the newly-created AR. The highest AR ID is 9999999. If the

highest ID is reached, a wraparound happens and the next unused ID, starting with 1, will be used.

For *qrsub*, the administrator and the user may define default request files (analogous to Altair Grid Engine_request for qsub), which can contain any of the possible command-line options.
A cluster-wide default request file is optional. If such a default request file is used, it must be placed under
$SGE_ROOT/$SGE_CELL/common/sge_ar_request (global defaults file).
A user-private default request file is optional. If it is used, it must be placed under
$HOME/.sge_ar_request (user private defaults file).

qralter can be used to change the attributes of an advance reservation. Whether an attribute can be changed depends on the state of the AR:

All attributes can be changed for ARs that are pending or running, but there must be no jobs running in the AR. It might become necessary to reschedule the AR, e.g. if resource requests (-l / -masterl) or queue requests (-q / -masterq) are modified. If rescheduling is not possible because the requested resources are not available in the given time frame, qralter will print an error message and the AR will not be modified.

If an AR is already running and has running jobs:

- simple modifications not affecting the reserved resources, such as modifying the name (-N) or the account string (-A), will always work
- modifying start time (-a), end time (-e), or duration (-d) will work, if the resources held by the AR will also be available in the new time frame. Reducing the time frame will always be accepted.
- if rescheduling of the AR would be necessary because e.g. resource requests will be modified (-l / -masterl) or the given set of resources will not be available in an extended time frame (-e / -d), qralter will print an error message and the AR will not be modified.

With the qmaster_param AR_DETACH_JOBS_ON_RESCHEDULE it is possible to have running jobs being detached from the AR and thus allowing re-scheduling; see sge_conf(5).

OPTIONS

-a date_time

Defines the activation (start) date and time of an AR. The option is not required. If it is omitted, the current date_time is assumed. Either a duration or end date_time must be specified for ARs. For details about date_time please see *sge_types*(1) For SRs the start time is optional and determines the earliest possible allocation time of an AR (but not the actual beginning of the first AR).

-A account_string

Identifies the account to which the resource reservation of the AR should be charged. For "**account_string**" value details please see the "**name**" definition in *sge_types*(1). In the ab-

sence of this parameter Altair Grid Engine will place the default account string “sge” in the accounting record of the AR.

-cal_week weekly_calendar

If a calendar is defined it is a Standing Reservation request. The weekly calendar has the same format as the calendar object (see *sge_calendar_conf*(5)) with the exception that only the “**on**” state is allowed. The weekly calendar defines the time ranges for which AR instances should be created. The scheduler tries to allocate a fixed amount of ARs. That amount can be influenced with the depth (“**-cal_depth**”) parameter. All ARs within an SR have the same ID. Jobs running within one AR are deleted at AR end (as with ARs) but queued jobs stay queued waiting for the next AR occurrence within the SR. Giving a start time, end time, or duration is optional for SRs. If given, the start time determines the earliest allowed start time for the first AR. The end time determines the last possible end time of the last AR within the SR. The duration specifies the end time. If one AR within the SR ends, new ARs are scheduled automatically, so that the total amount of allocated ARs are equal to the SR depth.

-cal_depth number

Specifies the depth of the SR. The depth determines how many ARs are allocated (scheduled) at any point in time in the future. ARs which cannot get enough resources are unallocated and not counted. If one AR ends, further ARs are allocated so that the depth is restored. If an AR cannot be allocated it will go into an Error state. Default for “**-cal_depth**” is 8.

-cal_jmp number

The jmp parameter determines how many of the first calendar entries can be jumped over if they cannot be allocated for ARs due to missing resources at SR submission time. If the first n ARs of a standing reservation cannot be allocated and “**-cal_jmp**” is less than n, the submission of the SR will be rejected. Default for “**-cal_jmp**” is 0.

-ckpt ckpt_name

Selects the checkpointing environment (see *sge_checkpoint*(5)) the AR jobs may request. Using this option guarantees that only queues providing this checkpoint environment will be reserved.

-d time

Defines the duration of the AR. The use of “**-d time**” is optional if “**-e date_time**” is requested. For details about “**time**” definition please see *sge_types*(1). For SRs the duration is optional and determines implicitly the end time, i.e. the last possible end time for the last AR within the SR.

-e date_time

Defines the end date and time of an AR. The use of "**-e date_time**" is optional if "**-d time**" is requested. For details about "**date_time**" definition please see *sge_types*(1). For SRs the end time is optional and determines the last possible end time for the last AR within the SR.

-fr y[es]|n[o]

Specifies the behavior of the AR at AR start time. With ("**-fr yes**") jobs still blocking resources being granted to the AR will be killed. With default behavior ("**-fr no**") jobs holding resources being granted to the AR can continue running and may prevent start of jobs in the AR. In the current implementation only resources requested by the AR via exclusive complexes are freed. The feature is suited for freeing whole queue instances or hosts or even the whole cluster, e.g. for maintenance purposes. Requesting "**-fr yes**" requires operator privileges.

-he y[es]|n[o]

Specifies the behavior when the AR goes into an error state. The AR goes into an error state when a reserved host goes into an unknown state, a queue error happens, or when a queue is disabled or suspended.

A hard error, "**-he yes**", means as long as the AR is in an error state no jobs using the AR will be scheduled. If a soft error, "**-he no**", is specified the AR remains usable with the remaining resources.
By default soft error handling is used.

-help

Prints a list of all options.

-version

Display version information for the command

-jc IGNORE_JC | NO_JC | ANY_JC |

If specified allows filtering of queue instances that may be considered for AR reservation.
By default, if the **-jc** switch is omitted or when **-jc IGNORE_JC** is specified, queues that accept JC **and** non-JC jobs at the same time will be considered for the AR. If the only queues that should be selected are those that do not allow execution of JC jobs, **-jc NO_JC** can be specified.
If the only queues that should be selected are those that allow execution of JC jobs, **-jc ANY_JC** is the right choice.
Additionally it is possible to filter queue instances that accept jobs derived from specific job classes. To achieve this a pattern for job class variants, or a specific job class variant name,

can be specified. This will select only those queue instances where jobs that are derived from a job class variant that matches the pattern are also allowed to be executed.

-l resource=value,...

Creates an AR in a Altair Grid Engine queue; the AR provides the given resource request list. *sge_complex*(5) describes how a list of available resources and their associated valid value specifiers can be obtained.
There may be multiple **-l** switches in a single command.

With AR submission it is possible to overwrite the consumable attribute of a resource description (the complex variable). Overwriting a consumable is only possible for complex variables which are already consumable (one of YES, JOB, HOST).

Syntax: -l resource[=value[{consumable}]][,resource[=value[{consumable}]],...] where consumable is one of YES, JOB, HOST.

Can be combined with **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

-masterl resource=value,...

Available only in combination with parallel jobs, i.e. together with the -pe option.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

qralter does allow changing the value of this option for the reservation; however the modification will only be effective if the job is not running in the reservation.

-masterq wc_queue_list

Available only in combination with parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may

make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qralter does allow changing the value of this option for the reservation; however the modification will only be effective if the job is not running in the reservation.

-m b|e|a|n

Defines or redefines under which circumstances mail is to be sent to the AR owner or to the users listed with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the AR
'e' Mail is sent at the end of the AR
'a' Mail is sent when the AR goes into an error state
'n' No mail is sent. This is the default for *qrsub*

-M user[@host],...

Defines or redefines the list of users to which the qmaster sends mail.

-masterq wc_queue_list

Only meaningful for a parallel AR request when used together with the -pe option.

This option is used to reserve the proper queues to match this request as if it were requested via qsub. A more detailed description of *wc_queue_list* can be found in *sge_types*(1).

-now y[es]|n[o]

This options impacts the queues selection for reservation.
With the "**-now y**" option, only queues with the qtype "INTERACTIVE" assigned will be considered for reservation. "**-now n**" is the default for *qrsub*.

-N name

The name of the AR. The name, if requested, must conform to "**name**" as defined in *sge_types*(1). Invalid names will be denied at submit time.

Note that names have to be unique when the qmaster param **SUBMIT_JOBS_WITH_AR_NAME** is set to true. qralter cannot change the AR name when using ar_name as an argument.

-P project_name

This option binds the AR to a project. Only members of this project are allowed to use this AR.

-w e|v

Specifies the validation level applied to the AR request.

The specifiers e and v define the following validation modes:

'v' Verify: does not submit the AR but prints an extensive validation report

'e' Error: rejects request if requirements cannot be fulfilled. This is the default for *qrsub*

-par allocation_rule

This option can be used with a parallel programming environment (PE).

It can be used to overwrite the allocation rule of the parallel environment into which an AR gets submitted with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel advance reservation.

This option can also be used after a **-petask** switch to overwrite the allocation rule for a specific set of tasks only. If there are multiple task sets defined at the command line, the scheduler will combine all groups with identical allocation rules. The tasks of such a group will be scheduled together. The scheduler will start with the group containing the task with the smallest ID, then will continue with the next group containing the smallest unscheduled task.

Valid allocation rules are **$pe_slots**, **$fill_up**, **$round_robin**, and a positive number as a **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

-pe parallel_env n[-[m]]|[-]m,...

Parallel programming environment (PE) to select for the AR queue reservation. Please see the details of a PE in *sge_pe*(5).

-petask tid_range_list ...

Available for *qrsub* and *qralter*.

Can be used to submit parallel advance reservations (submitted with **-pe** request); indicates that all resource requirements specified via **-q** and **-l**, and in combination with **-hard**, **-soft**, and **-par** that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the master task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request, the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The master task (ID 0) has a memory request of 4M. All slave tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

qralter can be used in combination with the **-petask** switch to completely replace the previously specified requests for an existing task ID range, as long as no additional restrictions

apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page. Please note also that attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission will be rejected.

-q wc_queue_list

Defines or redefines a list of cluster queues, queue domains, or queue instances that may be reserved by the AR. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-petask** to specify specific queue requirements for individual PE tasks or a group of PE tasks. Find more information in the corresponding sections of this man page.

-rao y[es]|n[o]

Specifies which resources will be considered for the AR at submission/reservation time. When using **-rao yes**, only currently available resources will be reserved. This means using only queues which are not in disabled, suspended, error or unknown state. The default setting for **-rao** is "no". A cluster-wide setting for only reserving available resources can be done via qmaster_param AR_RESERVE_AVAILABLE_ONLY; see *sge_conf*(5).

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-u [username|@access_list],...

Specifies the users allowed to submit jobs requesting the AR. The access is specified by a comma-separated list containing UNIX users or ACLs (see *sge_access_list*(5)). An ACL name is prefixed with an '@' sign.
By default only the AR owner is allowed to submit jobs requesting the AR.

Note: Only queues where all users specified in the list have access are considered for reservation (see *sge_queue_conf*(5)).

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell, *qsub*, *qsh*, *qlogin*, or *qalter* use (in order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the TCP port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead of defining the port.

FILES

$SGE_ROOT/$SGE_CELL/common/sge_ar_request

global defaults file

$HOME/.sge_ar_request

user-private defaults file

SEE ALSO

qrdel(1), *qrstat*(1), *qsub*(1), *sge_types*(1), *sge_checkpoint*(5), *sge_complex*(5), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_pe*(5), *sge_resource_quota*(5), *sge_calendar_conf*(5).

COPYRIGHT

qselect

NAME

qselect - select queues.

SYNTAX

qselect [**-help**] [**-l resource=val,...**] [**-pe pe_name,...**] [**-q wc_queue,...**] [**-s {r|p|s|z|hu|ho|hs|hj|ha|h}[+]**] [**-si session_id**] [**-U user,...**]

DESCRIPTION

qselect prints a list of Altair Grid Engine queue names corresponding to selection criteria specified in the *qselect* arguments described below. The output of *qselect* can be fed into other Altair Grid Engine commands to apply actions on the selected queue sets. For example together with the *-mqattr* option to *qconf*(1), *qselect* can be used to modify queue attributes on a set of queues.

OPTIONS

-help

Prints a listing of all options.

-version

Display version information for the *qselect* command.

-l resource[=value],...

Defines the resources to be granted by the queues which should be included in the queue list output. Matching is performed on queues based on non-mutable resource availability information only. That means load values are always ignored except the so-called static load values (i.e. “arch”, “num_proc”, “mem_total”, “swap_total” and “virtual_total”) ones. Also consumable utilization is ignored. If there are multiple -l resource requests they will be concatenated by a logical AND: a queue needs to offer all resources to be displayed.

-pe pe_name,...

Includes queues into the output which are attached to at least one of the parallel environments enlisted in the comma separated option argument.

-q wc_queue,...

Directly specifies the wildcard expression queue list to be included in the output. This option usually is only meaningful in conjunction with another *qselect* option to extract a subset of queue names from a list given by **-q**. Description of *wc_queue* can be found in *sge_types*(1).

-qs {a|c|d|o|s|u|A|C|D|E|S}

This option allows to filter for queue instances in certain states.

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-U user,...

Includes the queues to which the specified users have access in the *qselect* output.

-xml

This option can be used with all other options and changes the output to XML. The used schemas are referenced in the XML output. The output is printed to stdout.

-json

This option can be used with all other options and changes the output to JSON. The used schemas are referenced in the JSON output. The output is printed to stdout.

EXAMPLES

```
=====================================================
```

% qselect -l arch=linux
% qselect -l arch=linux -U andreas,shannon
% qconf -mattr queue h_vmem=1GB `qselect -l arch=linux`

==========

The first example prints the names of those queues residing on Linux machines. The second command in addition restricts the output to those queues with access permission for the users *andreas* and *shannon*. The third command changes the queue attribute *h_vmem* to 1 Gigabyte on queues residing on Linux machines (see the *qconf*(1) manual page for details on the *-mattr* option and the *sge_queue_conf*(5) manual page on details of queue configuration entries).

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qselect* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service “sge_qmaster” instead to define that port.

FILES

SGE_ROOT/SGE_CELL/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf*(1), *qmod*(1), *qstat*(1), *sge_queue_conf*(5), *sge_session_conf*(5)

COPYRIGHT

qsh

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request “m_mem_free” is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host (“m_mem_free_n<node>”).

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule “$pe_slots”. Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the “m_mem_free” host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule “$pe_slots”). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can’t fulfill the “m_mem_free” request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to “one core on the socket” and decrements the amount of memory on this socket (“m_mem_free_n<nodenumber>”). The global host memory “m_mem_free” on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from “m_mem_free” as well as from “m_mem_free_n<socketnumber>”, then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don’t offer “m_mem_free” memory are skipped (of course hosts that don’t offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory (“m_mem_free_n<N>”) and enough free cores, binds the job to these cores and sets memory allocation mode to “cores:strict” (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed (“m_mem_total_n<N>”). Binds the job to the complete socket, decrements the memory on that socket at “m_mem_free_n<N>” (as well as host globally on “m_mem_free”) and sets the memory allocation strategy to “cores” (preferred usage of socket local memory).

If the parameters request the “m_mem_free” complex, the resulting NUMA node memory requests can be seen in the “implicit_requests” row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

‘n’ none (default)
Switches off validation
‘e’ error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

qstat

NAME

qstat - show the status of Altair Grid Engine jobs and queues

SYNTAX

qstat [**-explain {a|A|c|E|m}[+]**] [[**-ext**] [**-f|-fjc**] [**-F** [**resource_name,...**]] [**-g {c|d|t}[+]**] [**-help**] [**-j [job_list]**] [**-nenv**] [**-l resource=val,...**] [**-ne**] [**-pe pe_name,...**] [**-ncb**] [**-njd**] [**-pri**] [**-q wc_queue_list**] [**-qs {a|c|d|o|s|u|A|C|D|E|S}**] [**-r**] [**-rr**] [**-s {r|p|s|z|S|N|P|hu|ho|hs|hd|hj|ha|h|a}[+]**] [**-si session_id**] [**-t**] [**-U user,...**] [**-u user,...**] [**-urg**] [**-xml**]

DESCRIPTION

qstat shows the current status of the available Altair Grid Engine queues, job classes and jobs. Selection options allow you to get information about specific jobs, job classes, queues or users. If multiple selections are done a queue or job class is only displayed if all selection criteria for a queue instance are met. Without any option *qstat* will display only a list of jobs with no queue or job class status information.

The administrator and the user may define files (see *sge_qstat*(5)), which can contain any of the options described below. A cluster-wide sge_qstat file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_qstat The user private file is searched at the location $HOME/.sge_qstat. The home directory request file has the highest precedence over the cluster global file. Command line can be used to override the flags contained in the files.

OPTIONS

-explain a|A|c|E|m

The character 'c' displays the reason for the c(onfiguration ambiguous) state of a queue instance or the c(onfiguration conflict) state of a job class variant. 'a' shows the reason for the alarm state. Suspend alarm state reasons will be displayed by 'A'. 'E' displays the reason for a queue instance error state. 'm' shows manager messages attached to queues when queue state changes were triggered (see **-msg** of *qmod*(1)).

The output format for the alarm reasons is one line per reason containing the resource value and threshold. For details about the resource value please refer to the description of the **Full Format** in section **OUTPUT FORMATS** below.

-ext

Displays additional information for each job related to the job ticket policy scheme (see OUTPUT FORMATS below).

-f

Specifies a “full” format display of information. The **-f** option causes summary information on all queues to be displayed along with the queued job list.

-fjc

Specifies a “job class” format display of information. The **-fjc** option causes summary information on all job class variants to be displayed along with a list of active jobs.

-F** [**resource_name,...]

Like in the case of **-f** information is displayed on all jobs as well as queues. In addition, *qstat* will present a detailed listing of the current resource availability per queue with respect to all resources (if the option argument is omitted) or with respect to those resources contained in the resource_name list. Please refer to the description of the **Full Format** in section **OUTPUT FORMATS** below for further detail.

-flt {pe|project} pe_list|project_list

Displays either all jobs belonging to the given pe_list or project_list. It is only possible to sort either by pe or by project.

-g {c|d|t}[+]

The **-g** option allows for controlling grouping of displayed objects.

With **-g c** a cluster queue summary is displayed. Find more information in the section **OUTPUT FORMATS**.

With **-g d** array jobs are displayed verbosely in a one line per job task fashion. By default, array jobs are grouped and all tasks with the same status (for pending tasks only) are displayed in a single line. The array job task id range field in the output (see section **OUTPUT FORMATS**) specifies the corresponding set of tasks.

With **-g t** parallel jobs are displayed verbosely in a one line per parallel job task fashion. By default, parallel job tasks are displayed in a single line. Also with **-g t** option the function of each parallel task is displayed rather than the jobs slot amount (see section **OUTPUT FORMATS**).

-help

Prints a listing of all options.

-version

Display version information for the *qstat* command.

-j [job_list]

Prints either for all pending jobs or the jobs contained in job_list various information. The job_list can contain job_ids, job_names, or wildcard expression *sge_types*(1).

For jobs in E(rror) state the error reason is displayed. For jobs that could not be dispatched during in the last scheduling interval the obstacles are shown, if 'schedd_job_info' in *sge_sched_conf*(5) is configured accordingly.

For running jobs available information on resource utilization is shown about consumed cpu time in seconds, integral memory usage in Gbytes seconds, amount of data transferred in Gbytes, I/O wait time in seconds, number of io operations, current virtual memory utilization in Mbytes, maximum virtual memory utilization in Mbytes, current resident set size, and maximum resident set size. This information is not available if resource utilization retrieval is not supported for the OS platform where the job is hosted. On Linux hosts additional memory values are reported when the execd param **ENABLE_MEM_DETAILS** is set to 1 (or true) (see also *sge_conf*(5)): pss (proportional set size), shared memory, private memory, and maxpss.

Only managers and operators can list jobs of other users if the manager adds the "-njd" switch to the sge_qstat file and "-u <user>" or "-u"*"" is specified. If the "-njd" switch is enabled the users only can view their own jobs.

If a job got a resource reservation information is shown about reservation start and end time, optionally the parallel environment the job got a reservation in, as well as the queue instances and the number of slots that were reserved for the job.

Please refer to the file <sge_root>/doc/load_parameters.asc for detailed information on the standard set of load values.

If the *sge_sched_conf*(5) parameter **schedd_job_info** is not set to **false**, the reasons why pending jobs could not be scheduled will be printed.

-jc jc_list

Specified a list of job class variant names to which job information is to be displayed. Find the definition of jc_list in *sge_types*(1).

-nenv

In combination with **-j job_list** the environment variables of the jobs are not requested.

-l resource[=value],...

Defines the resources required by the jobs or granted by the queues on which information is requested. Matching is performed on queues based on non-mutable resource availability information only. That means load values are always ignored except the so-called static load values (i.e. “arch”, “num_proc”, “mem_total”, “swap_total” and “virtual_total”). Consumable utilization is also ignored. The pending jobs are restricted to jobs that might run in one of the above queues. In a similar fashion also the queue-job matching bases only on non-mutable resource availability information. If there are multiple -l resource requests they will be concatenated by a logical AND: a queue needs to match all resources to be displayed.

-ne

In combination with **-f** the option suppresses the display of empty queues. This means all queues where actually no jobs are running are not displayed.

-ncb

In combination with **-ncb** the output of a command will suppress information of a requested binding and changes that have been applied to the topology string (real binding) for the host where the job is running. This information will disappear in combination with the parameters **-r** and **-j**.

Please note that this command line switch is intended to provide backward compatibility and will be removed in the next major release.

-njd

The **-njd** parameter can be used to disallow that normal users can see job details of jobs from other users with **qstat -j**. This switch can be added to the sge_qstat file by the admin user.

-pe pe_name,...

Displays status information with respect to queues which are attached to at least one of the parallel environments enlisted in the comma separated option argument. Status information for jobs is displayed either for those which execute in one of the selected queues or which are pending and might get scheduled to those queues in principle.

-pri

Displays additional information for each job related to the job priorities in general. (see OUTPUT FORMATS below).

-q wc_queue_list

Specifies a wildcard expression queue list to which job information is to be displayed. Find the definition of **wc_queue_list** in *sge_types*(1).

-qs {a|c|d|o|s|u|A|C|D|E|S}

Allows for the filtering of queue instances according to state.

-r

Prints extended information about the resource requirements of the displayed jobs.

Please refer to the **OUTPUT FORMATS** sub-section **Expanded Format** below for detailed information.

-rr

Prints job resource reservation information.

-s {p|r|s|z|S|N|P|hu|ho|hs|hd|hj|ha|h|a}[+]

Prints only jobs in the specified state, any combination of states is possible. **-s prsSNP** corresponds to the regular *qstat* output without **-s** at all. To show recently finished jobs, use **-s z**. To display jobs in user/operator/system/array-dependency hold, use the **-s hu/ho/hs/hd** option. To display preempted jobs in the S, N or P state use the **-s S/N/P** option. The **-s ha** option shows jobs which where submitted with the *qsub* **-a** command. *qstat* **-s hj** displays all jobs which are not eligible for execution unless the job has entries in the job dependency list. *qstat* **-s h** is an abbreviation for *qstat* **-s huhohshdhjha** and *qstat* **-s a** is an abbreviation for *qstat* **-s psrSNP** (see **-a**, **-hold_jid** and **-hold_jid_ad** options to *qsub*(1)).

-si session_id

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id** then such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-t

Prints extended information about the controlled sub-tasks of the displayed parallel jobs. Please refer to the **OUTPUT FORMATS** sub-section **Reduced Format** below for detailed information. Sub-tasks of parallel jobs should not be confused with array job tasks (see **-g** option above and **-t** option to *qsub*(1)).

-U user,...

Displays status information with respect to queues to which the specified users have access. Status information for jobs is displayed either for those which execute in one of the selected queues or which are pending and might get scheduled to those queues in principle.

-u user,...

Display information only on those jobs and queues being associated with the users from the given user list. Queue status information is displayed if the **-f** or **-F** options are specified additionally and if the user runs jobs in those queues.

The string **$user** is a placeholder for the current username. An asterisk "*" can be used as username wildcard to request any users' jobs be displayed. The default value for this switch is **-u $user**.

-urg

Displays additional information for each job related to the job urgency policy scheme (see OUTPUT FORMATS below).

-xml

This option can be used with all other options and changes the output to XML. The used schemas are referenced in the XML output. The output is printed to stdout. For more detailed information, the schemas for the qstat command can be found in $SGE_ROOT/util/resources/schemas/qstat.

If the **-xml** parameter is combined with **-ncb** then the XML output does not contain tags with information about job to core binding. You can also find schema files with the suffix "_ncb" in the directory $SGE_ROOT/util/resources/schemas/qstat that describe that changes.

-json

This option can be used with all other options and changes the output to JSON. The used schemas are referenced in the JSON output. The output is printed to stdout.

OUTPUT FORMATS

Depending on the presence or absence of the **-explain**, **-f**, **-F**, or **-qs** and **-r** and **-t** option three output formats need to be differentiated.

The **-ext** and **-urg** options may be used to display additional information for each job.

Cluster Queue Format (with -g c)

Following the header line a section for each cluster queue is provided. When queue instances selection are applied (-l -pe, -q, -U) the cluster format contains only cluster queues of the corresponding queue instances.

- the cluster queue name.
- an average of the normalized load average of all queue hosts. In order to reflect each hosts different significance the number of configured slots is used as a weighting factor when determining cluster queue load. Please note that only hosts with a np_load_value are considered for this value. When queue selection is applied only data about selected queues is considered in this formula. If the load value is not available at any of the hosts '-NA-' is printed instead of the value from the complex attribute definition.
- the number of currently used slots.
- the number of slots reserved in advance.
- the number of currently available slots.
- the total number of slots.
- the number of slots which is in at least one of the states 'aoACDS' and in none of the states 'cdsuE'
- the number of slots which are in one of these states or in any combination of them: 'cdsuE'
- the **-g c** option can be used in combination with **-ext**. In this case, additional columns are added to the output. Each column contains the slot count for one of the available queue states.

Reduced Format (without -f, -F, and -qs)

Following the header line a line is printed for each job consisting of

- the job ID.
- the priority of the job determining its position in the pending jobs list. The priority value is determined dynamically based on ticket and urgency policy set-up (see also *sge_priority*(5)).
- the name of the job.
- the user name of the job owner.
- the status of the job - one of d(eletion), E(rror), h(old), r(unning), R(estarted), s(uspended), S(uspended), e(N)hanced suspended, (P)reempted, t(ransfering), T(hreshold) or w(aiting).

The state d(eletion) indicates that a *qdel*(1) has been used to initiate job deletion. The states t(ransfering) and r(unning) indicate that a job is about to be executed or is already executing, whereas the states s(uspended), S(uspended), e(N)hanced suspended, (P)reempted and T(hreshold) show that an already running jobs has been suspended. The s(uspended) state is caused by suspending the job via the *qmod*(1) -s command, the S(uspended) state indicates either that the queue containing the job is suspended and therefore the job is also suspended or that a pending job got (S)uspended due to a preemptive action that was either triggered automatically by the system or by a manual preemption request triggered via *qmod*(1) -p ... S. e(N)hanced suspended and (P)reempted jobs in the pending jobs list are also preemptive states shown for pending jobs triggered either automatically or via *qmod*(1) -p ... N or -p ... P. T(hreshold) state shows that at least one suspend threshold of the corresponding queue was exceeded (see *sge_queue_conf*(5)) and that the job has been suspended as a consequence. The state R(estarted) indicates that the job was restarted. This can be caused by a job migration or because of one of the reasons described in the -r section of the *qsub*(1) command.

The states w(aiting) and h(old) only appear for pending jobs. The h(old) state indicates that a job currently is not eligible for execution due to a hold state assigned to it via *qhold*(1), *qalter*(1) or the *qsub*(1) **-h** option or that the job is waiting for completion of the jobs to which job dependencies have been assigned to the job via the **-hold_jid** or **-hold_jid-ad** options of *qsub*(1) or *qalter*(1).

The state E(rror) appears for pending jobs that couldn't be started due to job properties. The reason for the job error is shown by the *qstat*(1) **-j job_list** option.

- the submission or start time and date of the job.
- the queue the job is assigned to (for running or suspended jobs only).
- the number of job slots or the function of parallel job tasks if **-g t** is specified.

Without **-g t** option the total number of slots occupied resp. requested by the job is displayed. For pending parallel jobs with a PE slot range request, the assumed future slot allocation is displayed. With **-g t** option the function of the running jobs (MASTER or SLAVE - the latter for parallel jobs only) is displayed.

- the array job task id. Will be empty for non-array jobs. See the **-t** option to *qsub*(1) and the **-g** above for additional information.

If the **-t** option is supplied, each status line always contains parallel job task information as if **-g t** were specified and each line contains the following parallel job subtask information:

- the parallel task ID (do not confuse parallel tasks with array job tasks),
- the status of the parallel task - one of r(unning), R(estarted), s(uspended), S(uspended), T(hreshold), w(aiting), h(old), or x(exited).
- the cpu, memory, and I/O usage,
- the exit status of the parallel task,
- and the failure code and message for the parallel task.

Full Format (with -f and -F)

Following the header line a section for each queue separated by a horizontal line is provided. For each queue the information printed consists of

- the queue name,
- the queue type - one of B(atch), I(nteractive), C(heckpointing), P(arallel) or combinations thereof or N(one),
- the number of used and available job slots,
- the load average of the queue host,
- the architecture of the queue host and
- the state of the queue - one of u(nknown) if the corresponding *sge_execd*(8) cannot be contacted, a(larm), A(larm), B(locked on SSOS), C(alendar suspended), s(uspended), S(ubordinate), d(isabled), D(isabled), E(rror), (un)L(icensed) or combinations thereof.

If the state is a(larm) at least on of the load thresholds defined in the *load_thresholds* list of the queue configuration (see *sge_queue_conf*(5)) is currently exceeded, which prevents from scheduling further jobs to that queue.

As opposed to this, the state A(larm) indicates that at least one of the suspend thresholds of the queue (see *sge_queue_conf*(5)) is currently exceeded. This will result in jobs running in that queue being successively suspended until no threshold is violated.

The states s(uspended) and d(isabled) can be assigned to queues and released via the *qmod*(1) command. Suspending a queue will cause all jobs executing in that queue to be suspended.

The states D(isabled), (un)L(icensed) and C(alendar suspended) indicate that the queue has been disabled manually, automatically due to software license restrictions or suspended automatically via the calendar facility of Altair Grid Engine (see *sge_calendar_conf*(5)), while the S(ubordinate) state indicates, that the queue has been suspend via subordination to another queue (see *sge_queue_conf*(5) for details). When suspending a queue (regardless of the cause) all jobs executing in that queue are suspended too.

The state B(locked on SSOS) indicates that this subordinate queue is blocked from running any new jobs as the threshold for the combined number of jobs on the parent and child queues has been reached.

If an E(rror) state is displayed for a queue, *sge_execd*(8) on that host was unable to locate the *sge_shepherd*(8) executable on that host in order to start a job. Please check the error logfile of that *sge_execd*(8) for leads on how to resolve the problem. Please enable the queue afterwards via the **-c** option of the *qmod*(1) command manually.

If the c(onfiguration ambiguous) state is displayed for a queue instance this indicates that the configuration specified for this queue instance in *sge_conf*(5) is ambiguous. This state is cleared when the configuration becomes unambiguous again. This state prevents further jobs from being scheduled to that queue instance. Detailed reasons why a queue instance entered the c(onfiguration ambiguous) state can be found in the *sge_qmaster*(8) messages

file and are shown by the qstat -explain switch. For queue instances in this state the cluster queue's default settings are used for the ambiguous attribute.

If an o(rphaned) state is displayed for a queue instance, it indicates that the queue instance is no longer demanded by the current cluster queue's configuration or the host group configuration. The queue instance is kept because jobs which not yet finished jobs are still associated with it, and it will vanish from qstat output when these jobs have finished. To quicken vanishing of an orphaned queue instance associated job(s) can be deleted using *qdel*(1). A queue instance in (o)rphaned state can be revived by changing the cluster queue configuration accordingly to cover that queue instance. This state prevents from scheduling further jobs to that queue instance.

If the **-F** option was used, resource availability information is printed following the queue status line. For each resource (as selected in an option argument to **-F** or for all resources if the option argument was omitted) a single line is displayed with the following format:

- a one letter specifier indicating whether the current resource availability value was dominated by either

'**g**' - a cluster global,
'**h**' - a host total or
'**q**' - a queue related resource consumption.

- a second one letter specifier indicating the source for the current resource availability value, being one of

'**l**' - a load value reported for the resource,
'**L**' - a load value for the resource after administrator defined load scaling has been applied,
'**c**' - availability derived from the consumable resources facility (see *sge_complex*(5)),
'**f**' - a fixed availability definition derived from a non-consumable complex attribute or a fixed resource limit.

- after a colon the name of the resource on which information is displayed.
- after an equal sign the current resource availability value.

The displayed availability values and the sources from which they derive are always the minimum values of all possible combinations. Hence, for example, a line of the form "qf:h_vmem=4G" indicates that a queue currently has a maximum availability in virtual memory of 4 Gigabyte, where this value is a fixed value (e.g. a resource limit in the queue configuration) and it is queue dominated, i.e. the host in total may have more virtual memory available than this, but the queue doesn't allow for more. Contrarily a line "hl:h_vmem=4G" would also indicate an upper bound of 4 Gigabyte virtual memory availability, but the limit would be derived from a load value currently reported for the host. So while the queue might allow for jobs with higher virtual memory requirements, the host on which this particular queue resides currently only has 4 Gigabyte available.

If there are additional resources that are available through preempted jobs then the resource amount is shown in the form "+1". This additional output is optional.

If positive or negative affinity is defined for one or more resources that are consumed by jobs running on a hosts or queue then the corresponding affinity values will be shown in the form "(haff=1.000000, qaff=1.000000)" additionally. haff shows the internal host affinity values whereas qaff shows the queue affinity values.

A string like "qc:slots=2+1 (haff=5.000000, qaff=5.000000)" shows the amount of resources of the slots queue consumable. In total there are 3 (one of those slots due to a preempted job). Host and queue affinity for slots is 5.0.

If the **-explain** option was used with the character 'a' or 'A', information about resources is displayed, that violate load or suspend thresholds.
The same format as with the **-F** option is used with following extensions:

- the line starts with the keyword 'alarm'
- appended to the resource value is the type and value of the appropriate threshold

After the queue status line (in case of **-f**) or the resource availability information (in case of **-F**) a single line is printed for each job running currently in this queue. Each job status line contains

- the job ID,
- the priority of the job determining its position in the pending jobs list. The priority value is determined dynamically based on ticket and urgency policy set-up (see also *sge_priority*(5)).
- the job name,
- the job owner name,
- the status of the job - one of t(ransfering), r(unning), R(estarted), s(uspended), S(uspended) or T(hreshold) (see the **Reduced Format** section for detailed information),
- the submission or start time and date of the job.
- the number of job slots or the function of parallel job tasks if **-g t** is specified.

Without **-g t** option the number of slots occupied per queue resp. requested by the job is displayed. For pending parallel jobs with a PE slot range request, the assumed future slot allocation is displayed. With **-g t** option the function of the running jobs (MASTER or SLAVE - the latter for parallel jobs only) is displayed.

If the **-t** option is supplied, each job status line also contains

- the task ID,
- the status of the task - one of r(unning), R(estarted), s(uspended), S(uspended), T(hreshold), w(aiting), h(old), or x(exited) (see the **Reduced Format** section for detailed information),
- the cpu, memory, and I/O usage,

- the exit status of the task,
- and the failure code and message for the task.

Following the list of queue sections a *PENDING JOBS* list may be printed in case jobs are waiting for being assigned to a queue. A status line for each waiting job is displayed being similar to the one for the running jobs. The differences are that the status for the jobs is w(aiting) or h(old), that the submit time and date is shown instead of the start time and that no function is displayed for the jobs.

In very rare cases, e.g. if *sge_qmaster*(8) starts up from an inconsistent state in the job or queue spool files or if the **clean queue** (**-cq**) option of *qconf*(1) is used, *qstat* cannot assign jobs to either the running or pending jobs section of the output. In this case as job status inconsistency (e.g. a job has a running status but is not assigned to a queue) has been detected. Such jobs are printed in an *ERROR JOBS* section at the very end of the output. The ERROR JOBS section should disappear upon restart of *sge_qmaster*(8). Please contact your Altair Grid Engine support representative if you feel uncertain about the cause or effects of such jobs.

Job Class Format (with -fjc)

Following the header line a section for a job class variant separated by a horizontal line is provided. For each job class variant the information printed consist of

- the job class variant name,
- the ownership (O) - 'X' will be displayed if the executing user is a owner of the job class
- the usability (U) - 'X' will be displayed if the executing user is allowed to derive new jobs from the corresponding job class variant.
- the states - d(isabled) if the job class variant cannot be used to create new jobs. c(onfiguration conflict) if the queue is in conflict with the specification of the template job class. o(rphaned) when a job class variant is in process of deletion.

Details concerning the different states can be found in *sge_job_class*((5))

Expanded Format (with -r)

If the **-r** option was specified together with *qstat*, the following information for each displayed job is printed (a single line for each of the following job characteristics):

- The job and master queue name.
- The hard and soft resource requirements of the job as specified with the *qsub*(1) **-l** option. The per resource addend when determining the jobs urgency contribution value is printed (see also *sge_priority*(5)).
- The requested parallel environment including the desired queue slot range (see **-pe** option of *qsub*(1)).

- The requested checkpointing environment of the job (see the *qsub*(1) **-ckpt** option).
- In case of running jobs, the granted parallel environment with the granted number of queue slots.
- The requested job binding parameters.

Enhanced Output (with -ext)

For each job the following additional items are displayed:

ntckts

The total number of tickets in normalized fashion.

project

The project to which the job is assigned as specified in the *qsub*(1) **-P** option.

department

The department, to which the user belongs (use the **-sul** and **-su** options of *qconf*(1) to display the current department definitions).

cpu

The current accumulated CPU usage of the job in seconds.

mem

The current accumulated memory usage of the job in Gbytes seconds.

io

The current accumulated IO usage of the job.

tckts

The total number of tickets assigned to the job currently

ovrts

The override tickets as assigned by the **-ot** option of *qalter*(1).

otckt

The override portion of the total number of tickets assigned to the job currently

ftckt

The functional portion of the total number of tickets assigned to the job currently

stckt

The share portion of the total number of tickets assigned to the job currently

share

The share of the total system to which the job is entitled currently.

Enhanced Output (with -urg)

For each job the following additional urgency policy related items are displayed (see also *sge_priority*(5)):

nurg

The jobs total urgency value in normalized fashion.

urg

The jobs total urgency value.

rrcontr

The urgency value contribution that reflects the urgency that is related to the jobs overall resource requirement.

wtcontr

The urgency value contribution that reflects the urgency related to the jobs waiting time.

dlcontr

The urgency value contribution that reflects the urgency related to the jobs deadline initiation time.

deadline

The deadline initiation time of the job as specified with the *qsub*(1) **-dl** option.

Enhanced Output (with -pri)

For each job, the following additional job priority related items are displayed (see also *sge_priority*(5)):

nurg

The job's total urgency value in normalized fashion.

npprior

The job's **-p** priority in normalized fashion.

ntckts

The job's ticket amount in normalized fashion.

ppri

The job's **-p** priority as specified by the user.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qstat* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SGE_LONG_QNAMES

Qstat does display queue names up to 30 characters. If that is too much or not enough, one can set a custom length with this variable. The minimum display length is 10 characters. If one does not know the best display length, one can set SGE_LONG_QNAMES to -1 and qstat will figure out the best length.

SGE_LONG_JOB_NAMES

Qstat does display job names up to 10 characters. If that is not enough, one can set a custom length with this variable. The minimum display length is 10 characters. If one does not know the best display length, one can set SGE_LONG_JOB_NAMES to -1 and qstat will figure out the best length.

FILES

<sge_root>/<cell>/common/act_qmaster

Altair Grid Engine master host file

<sge_root>/<cell>/common/sge_qstat

cluster qstat default options

$HOME/.sge_qstat

user qstat default options

SEE ALSO

sge_intro(1), *qalter*(1), *qconf*(1), *qhold*(1), *qhost*(1), *qmod*(1), *qsub*(1), *sge_queue_conf*(5), *sge_session_conf*(5), *sge_execd*(8), *sge_qmaster*(8), *sge_shepherd*(8).

COPYRIGHT

qsub

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request "m_mem_free" is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host ("m_mem_free_n<node>").

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule "$pe_slots". Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the "m_mem_free" host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule "$pe_slots"). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can't fulfill the "m_mem_free" request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to "one core on the socket" and decrements the amount of memory on this socket ("m_mem_free_n<nodenumber>"). The global host memory "m_mem_free" on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from "m_mem_free" as well as from "m_mem_free_n<socketnumber>", then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don't offer "m_mem_free" memory are skipped (of course hosts that don't offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory ("m_mem_free_n<N>") and enough free cores, binds the job to these cores and sets memory allocation mode to "cores:strict" (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed ("m_mem_total_n<N>"). Binds the job to the complete socket, decrements the memory on that socket at "m_mem_free_n<N>" (as well as host globally on "m_mem_free") and sets the memory allocation strategy to "cores" (preferred usage of socket local memory).

If the parameters request the "m_mem_free" complex, the resulting NUMA node memory requests can be seen in the "implicit_requests" row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

‘n’ none (default)
Switches off validation
‘e’ error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

sge_ckpt

NAME

Altair Grid Engine Checkpointing - the Altair Grid Engine checkpointing mechanism and checkpointing support

DESCRIPTION

Altair Grid Engine supports two levels of checkpointing: the user level and a operating system provided transparent level. User level checkpointing refers to applications, which do their own checkpointing by writing restart files at certain times or algorithmic steps and by properly processing these restart files when restarted.

Transparent checkpointing has to be provided by the operating system and is usually integrated in the operating system kernel. An example for a kernel integrated checkpointing facility is the Hibernator package from Softway for SGI IRIX platforms.

Checkpointing jobs need to be identified to the Altair Grid Engine system by using the *-ckpt* option of the *qsub*(1) command. The argument to this flag refers to a so called checkpointing environment, which defines the attributes of the checkpointing method to be used (see *sge_checkpoint*(5) for details). Checkpointing environments are setup by the *qconf*(1) options *-ackpt*, *-dckpt*, *-mckpt* and *-sckpt*. The *qsub*(1) option *-c* can be used to overwrite the *when* attribute for the referenced checkpointing environment.

If a queue is of the type CHECKPOINTING, jobs need to have the checkpointing attribute flagged (see the *-ckpt* option to *qsub*(1)) to be permitted to run in such a queue. As opposed to the behavior for regular batch jobs, checkpointing jobs are aborted under conditions, for which batch or interactive jobs are suspended or even stay unaffected. These conditions are:

- Explicit suspension of the queue or job via *qmod*(1) by the cluster administration or a queue owner if the *x* occasion specifier (see *qsub*(1) *-c* and *sge_checkpoint*(5)) was assigned to the job.
- A load average value exceeding the suspend threshold as configured for the corresponding queues (see *sge_queue_conf*(5)).
- Shutdown of the Altair Grid Engine execution daemon *sge_execd*(8) being responsible for the checkpointing job.

After abortion, the jobs will migrate to other queues unless they were submitted to one specific queue by an explicit user request. The migration of jobs leads to a dynamic load balancing. **Note:** The abortion of checkpointed jobs will free all resources (memory, swap space) which the job occupies at that time. This is opposed to the situation for suspended regular jobs, which still cover swap space.

RESTRICTIONS

When a job migrates to a queue on another machine at present no files are transferred automatically to that machine. This means that all files which are used throughout the entire job including restart files, executables and scratch files must be visible or transferred explicitly (e.g. at the beginning of the job script).

There are also some practical limitations regarding use of disk space for transparently checkpointing jobs. Checkpoints of a transparently checkpointed application are usually stored in a checkpoint file or directory by the operating system. The file or directory contains all the text, data, and stack space for the process, along with some additional control information. This means jobs which use a very large virtual address space will generate very large checkpoint files. Also the workstations on which the jobs will actually execute may have little free disk space. Thus it is not always possible to transfer a transparent checkpointing job to a machine, even though that machine is idle. Since large virtual memory jobs must wait for a machine that is both idle, and has a sufficient amount of free disk space, such jobs may suffer long turnaround times.

SEE ALSO

sge_intro(1), *qconf*(1), *qmod*(1), *qsub*(1), *sge_checkpoint*(5), *Altair Grid Engine Installation and Administration Guide*, *Altair Grid Engine User's Guide*

COPYRIGHT

sge_intro

NAME

Altair Grid Engine - a facility for executing UNIX jobs on remote machines

DESCRIPTION

Altair Grid Engine is a facility for executing UNIX batch jobs (shell scripts) on a pool of cooperating workstations. Jobs are queued and executed remotely on workstations at times when those workstations would otherwise be idle or only lightly loaded. The work load is distributed among the workstations in the cluster corresponding to the load situation of each machine and the resource requirements of the jobs.

User level checkpointing programs are supported and a transparent checkpointing mechanism is provided (see *sge_ckpt*(1)). Checkpointing jobs migrate from workstation to workstation without user intervention on load demand. In addition to batch jobs, interactive jobs and parallel jobs can also be submitted to Altair Grid Engine.

USER INTERFACE

The Altair Grid Engine user interface consists of several programs which are described separately.

qacct(1)
qacct extracts arbitrary accounting information from the cluster logfile.

qalter(1)
qalter changes the characteristics of already submitted jobs.

qconf(1)
qconf provides the user interface for configuring, modifying, deleting and querying queues and the cluster configuration.

qdel(1)
qdel provides the means for a user/operator/manager to cancel jobs.

qhold(1)
qhold holds back submitted jobs from execution.

qhost(1)
qhost displays status information about Altair Grid Engine execution hosts.

qlogin(1)
qlogin initiates a telnet or similar login session with automatic selection of a low loaded and suitable host.

qmake(1)
qmake is a replacement for the standard Unix *make* facility. It extends make by its ability to distribute independent make steps across a cluster of suitable machines.

qmod(1)
qmod allows the owner(s) of a queue to suspend and enable all queues associated with his machine (all currently active processes in this queue are also signaled) or to suspend and enable jobs executing in the owned queues.

qmon(1)
qmon provides a Motif command interface to all Altair Grid Engine functions. The status of all or a private selection of the configured queues is displayed on-line by changing colors at corresponding queue icons.

qquota(1)
qquota provides a status listing of all currently used resource quotas (see *sge_resource_quota*(5).)

qresub(1)
qresub creates new jobs by copying currently running or pending jobs.

qrls(1)
qrls releases holds from jobs previously assigned to them e.g. via *qhold*(1) (see above).

qrdel(1)
qrdel provides the means to cancel advance reservations.

qrsh(1)
qrsh can be used for various purposes such as providing remote execution of interactive applications via Altair Grid Engine comparable to the standard Unix facility rsh, to allow for the submission of batch jobs which, upon execution, support terminal I/O (standard/error output and standard input) and terminal control, to provide a batch job submission client which remains active until the job has finished or to allow for the Altair Grid Engine-controlled remote execution of the tasks of parallel jobs.

qrstat(1)
qrstat provides a status listing of all advance reservations in the cluster.

qrsub(1)
qrsub is the user interface for submitting a advance reservation to Altair Grid Engine.

qselect(1)
qselect prints a list of queue names corresponding to specified selection criteria. The output of *qselect* is usually fed into other Altair Grid Engine commands to apply actions on a selected set of queues.

qsh(1)
qsh opens an interactive shell (in an *xterm* (1)) on a low loaded host. Any kind of interactive jobs can be run in this shell.

qstat(1)
qstat provides a status listing of all jobs and queues associated with the cluster.

qsub(1)
qsub is the user interface for submitting a job to Altair Grid Engine.

SEE ALSO

sge_ckpt(1), *qacct*(1), *qalter*(1), *qconf*(1), *qdel*(1), *qhold*(1), *qhost*(1), *qlogin*(1), *qmake*(1), *qmod*(1), *qmon*(1), *qresub*(1), *qrls*(1), *qrsh*(1), *qselect*(1), *qsh*(1), *qstat*(1), *qsub*(1), *Altair Grid Engine Installation Guide*, *Altair Grid Engine Administration Guide*, *Altair Grid Engine User's Guide*.

COPYRIGHT

sge_preemption

NAME

Preemption - Manual, Semi-Automatic and Automatic Preemption in Altair Grid Engine

DESCRIPTION

Altair Grid Engine clusters can cope with different types of workloads. The configuration of the scheduler component defines the way how to handle different workloads in the daily operation. Various policies can be combined to reflect the requirements.

In previous versions of Grid Engine enforcing policies sometimes was difficult especially when high priority jobs would require resources of lower priority jobs that already bind resources like slots, memory or licenses. In such cases it was required to use slot-wise suspend on subordinate to make such resources available or reservation and advance reservation functionality could be used to reserve resources for such high priority jobs before they drop in.

Altair Grid Engine 8.3 (and above) additionally provide the possibility to enforce configured policies when required resources are already in use. This can be done through preemption. This document describes preemptive scheduling as an addition to the Altair Grid Engine job handling and scheduling that makes it possible to more closely follow the goals defined by the policies and if necessary enforce them.

TERMS

Following paragraphs describe a couple of terms that are used throughout this document.

Jobs which have high priority based on the configured policies can get the role of an *preemption consumer* that can cause a *preemption action* to be performed for one or more running jobs that have the role of an *preemption provider*. In general all those running jobs are considered as *preemption provider* where the priority is smaller than that of the *preemption consumer*.

There are different *preemption actions* available in Altair Grid Engine. What all of them have in common is that they will make all or a subset of the bound resources of a *preemption provider* available so that they can be used by one or more *preemption consumer*. Different *preemption actions* differ in the way how bound resources are freed and how the Altair Grid Engine system will make the bound resources available.

Preemption actions can be executed by Altair Grid Engine due to three different *preemption triggers*. A *preemption trigger* will define the time and has an influence on the chosen *preemption action* that is performed. In general *preemption trigger* can be *manual*, *semi-automatic* or *automatic*.

A *preemption consumer* that consumes resources that got available through triggering a *preemption action* has the role on an *preemptor* whereas those jobs that get forced to free resources are considered as *preemptee*.

Please note: Within Altair Grid Engine 8.3 *manual preemption* is implemented. *semi-automatic* or *automatic* trigger will follow with upcoming releases.

PREEMPTIVE TRIGGER AND ACTIONS

Altair Grid Engine 8.3 provides six different preemption actions to preempt a job. With manual preemption the user/manager has to choose which of the available preemptive actions should be used to trigger preemption of a job. With semi-automatic and automatic preemption mechanisms (available with future versions of Altair Grid Engine) either the system configuration or the Altair Grid Engine scheduler decides automatically which preemption action will be taken to release resources.

The six preemptive actions differ in the way which of the resources will be available for other jobs after the preemptee got preempted. Some of those actions have restrictions on which job types they can be applied as well as who is allowed to trigger them. The actions differ also in the way how they treat the processes that are executed on behalf of a job that gets preempted.

Within Altair Grid Engine all preemptive actions are represented by single capital letter (**T**, **R**, **C**, **P**, **N** or **S**) that is either passed to a command, specified in a configuration object or that is shown in command output to show the internal state of a job.

Some of the preemptive actions trigger the *suspend_method* that might be defined in the queue where the preemptee is executed. To be able to distinguish different preemption actions within the *suspend_method* an optional argument named *$action* might be used as pseudo argument when the method is defined. That argument will be expanded to the corresponding letter that represents the preemptive action during runtime.

(**T**)erminate Action: The preemptee will be terminated. As soon as all underlying processes are terminated all resources that were bound by that preemptee will be reported as free. The T-action can be applied to any job. Users can apply it only to own jobs.

(**C**)heckpoint Action: The preemptee will be checkpointed. As soon as a checkpoint is written and all underlying processes are terminated all bound resources will be reported as available and the job will be rescheduled. This preemption action can only be applied to checkpointing jobs where a checkpointing environment was specified during submission of this job.

(**R**)erun Action: The preempted job will be rescheduled. As soon as all underlying processes are terminated all bound resources will be reported as available. Managers can enforce the rerun of jobs even if those jobs are not tagged as rerun-able on the job or queue level.

(**P**)reemption Action: The preemptee will be preempted. Preempted means that the configured *queue-suspend* method (*$action* set to *P*) will be executed that might trigger additional operations to notify the processes about the upcoming preemption so that those processes can release bound resources by itself. After that the processes are suspended and all consumable resources, where the attribute available-after-preemption (aapre) is set to true, are reported as free. Not-available-after-preemption resources are still reported to

be bound by the preempted job. The preemption action can be applied to all preemption providers whereas users can only preempt own jobs.

e(**N**)hanced Suspend Action: Similar to the preempt action the queue *suspend_method* (*$action* set to "*E*") will be triggered before the preemptee gets suspended. Only non-memory-based consumables (including LO-managed license resources) are reported as free when the processes are suspended. Memory-based consumables that are available-after-preemption and also not-available-after-preemption consumables will still be reported as bound by the enhanced suspended job. This preemption action can be applied to all preemption providers. Users can only preempt own jobs.

(**S**)uspend Action: Similar to the preempt action the triggered method will be the *suspend_method* (*$action* set to "*S*") before the preemptee gets suspended. Only consumed slots (and LO-managed license resources) will be available after suspension. All other resources, independent if they are tagged as available-after-preemption or not-available-after-preemption in the complex configuration, will be reported as still in use. This preemption action can be applied to all preemption providers. Users can only preempt own jobs.

Which of the six preemptive action should be chosen to manually preempt a job? If a job is checkpointable then it should be the **C**-action. Here all consumed resources of the preemptee will be available for higher priority jobs. The preemptee can continue its work at that point where the last checkpoint was written when it is restarted.

Although also the **T**-action and the **R**-action provide the full set of resources but they should be seen as the last resort when no less disruptive preemptive actions can be applied. Reason for this is that the computational work of the preemptee up to the point in time where the preemptee is rescheduled or terminated might get completely lost which would be a waste of resources.

From the Altair Grid Engine perspective also the P-action makes all bound resources (slots + memory + other consumable resources where *aapree* of the complex is set to *true*) available for higher priority jobs. But this is only correct if the machine has enough swap space configured so that the underlying OS is able to move consumed physical memory pages of the suspended processes into that swap space and also when the application either releases consumed resources (like software licenses, special devices, ...) automatically or when a *suspend_method* can be configured to trigger the release of those resources. The **N**-action can be used for jobs that run on hosts without or with less configured swap space. It will make only non-memory-based consumables available (slots + other consumable resources where *aapree* of the complex is set to *true*).

If jobs either do not use other resources (like software licenses, special devices, ...) and when memory consumption is not of interest in the cluster, then the **S**-action can be chosen. It is the simplest preemptive action that provides slots (and LO-licenses) only after preemption. Please note that the S-action and S-state of jobs is different from the s-state of a job (triggered via *qmod* -s command). A regularely suspended job does not release slots of that job. Those slots are blocked by the manually suspended job.

The **P** and **N**-action will make consumable resources of preemptees available for higher priority jobs. This will be done automatically for all preconfigured consumable resources in a cluster. For those complexes the available-after-preemption-attribute (*aapre*) is set to *YES*. Managers of a cluster can change this for predefined complexes. They also have to decide

if a self-defined resource gets available after preemption. For Resources that should be ignored by the preemptive scheduling functionality the *aapre*-attribute can be set to *NO*.

Please note that the resource set for each explained preemptive action defines the maximum set of resources that might get available through that preemption action. Additional scheduling parameters (like *prioritize_preemptees* or *preemptees_keep_resources* that are further explained below) might reduce the resource set that get available through preemption to a subset (only those resources that are demanded by a specified *preemption_consumer*) of the maximum set.

MANUAL PREEMPTION

Manual preemption can be triggered with the *qmod* command in combination with the *p*-switch. The *p*-switch expects one job ID of a *preemption_consumer* followed by one or multiple job ID's or job names of *preemption_provider*. As last argument the command allows to specify a character representing one of the six *preemptive_actions*. This last argument is optional. *P*-action will be used as default if the argument is omitted.

Syntax:

```
qmod [-f] -p <preemption_consumer>
          <preemption_provider> [<preemption_provider> ...]
          [<preemption_action>]

<preemption_consumer> := <job_ID> .
<preemption_provider> := <job_ID> | <job_name> .
<preemption_action>   := "P" | "N" | "S" | "C" | "R" | "T" .
```

The manual preemption request will only be accepted if it is valid. Manual preemption request will be rejected when:

- Resource reservation is disabled in the cluster.
- *preemption_consumer* has no reservation request.
- At least one specified *preemption_provider* is not running.
- **C**-action is requested but there is at least one *preemption_provider* that is not checkpointable.
- **R**-action is requested but there is at least one *preemption_provider* that is neither tagged as rerunnable nor the queue where the job is running is a rerunnable queue. (Manager can enforce the R-action in combination with the *f*-switch).

Manual preemption requests are not immediately executed after they have been accepted by the system. The Altair Grid Engine scheduler is responsible to trigger manual preemption during the next scheduling run. Preemption will only be triggered if the resources will not otherwise be available to start the preemption consumer within a configurable time frame (see *preemption_distance* below). If enough resources are available or when the scheduler

sees that they will be available in near future then the manual preemption request will be ignored.

Please note that resources that get available through preemption are only reserved for the specified *preemption_consumer* if there are no other jobs of higher priority that also demands those resources. If there are jobs of higher priority then those jobs will get the resources and the specified *preemption_consumer* might stay in pending state till either the higher priority jobs leaves the system or another manual preemption request is triggered.

Preemptees will automatically trigger a reservation of those resources that they have lost due to preemption. This means that they can be reactivated as soon as they are eligible due to its priority and as soon as the missing resources get available. There is no dependency between a preemptor and the preemptees. All or a subset of preemptees might get restarted even if the preemptor is still running if demanded resources are added to the cluster or get available due to the job end of other jobs.

Preemptees will have the jobs state **P**, **N** or **S** (shown in the *qstat* output or *qmon* dialogs) depending of the corresponding preemption action that was triggered. Those jobs, as well as preemptees that get rescheduled due to the **R**-action, will appear as pending jobs even if they still hold some resources. Please note that regularly suspended jobs (in **s**-state due to *qmod* -s) still consume all resources and therefore block the queue slots for other jobs. *qstat* -j command can be used to see which resources are still bound by preemptees.

PREEMPTION CONFIGURATION

Following scheduling configuration parameters are available to influence the preemptive scheduling as well as the preemption behaviour of the Altair Grid Engine cluster. They

max_preemptees: The maximum number of preemptees in the cluster. As preempted jobs might hold some resources (e.g memory) and through the *preemptees_keep_resources* parameter might even hold most of their resources a high number of preemptees can significantly impact cluster operation. Limiting the number of preemptees will limit the amount of held but unused resources.

prioritize_preemptees: By setting this parameter to *true* or *1* preemptees get a reservation before the regular scheduling is done. This can be used to ensure that preemptees get restarted again at latest when the preemptor finishes, unless resources required by the preemptee are still held by jobs which got backfilled. *prioritize_preemptees* in combination with disabling of backfilling provides a guarantee that preemptees get restarted at least when the preemptor finishes, at the expense of lower cluster utilization.

preemptees_keep_resources: When a job gets preempted only those resources will get freed which will be consumed by the preemptor.This prevents resources of a preemptee from getting consumed by other jobs. *prioritize_preemptees* and *preemptees_keep_resources* in combination provide a guarantee that preemptees get restarted at latest when the preemptor finishes, at the expense of a waste of resources and bad cluster utilization. Exception: Licenses managed through LO and a license manager cannot be held by a preemptee. As the preemptee processes will be suspended the license manager might assume the license to be free which will lead to the license be consumed by a different job. When the preemptee processes get unsuspended again a license query would fail if the license is held.

preemption_distance: A preemption will only be triggered if the resource reservation that could be done for a job is farther in the future than the given time interval (hh:mm:ss, default 00:15:00). Reservation can be disabled by setting the value to 00:00:00. Reservation will also be omitted if preemption of jobs is forced by a manager manually using (via *qmod* -f -p ...).

PREEMPTION IN COMBINATION WITH LICENSE ORCHESTRATOR

License complexes that are reported by License Orchestrator are automatically defined as available-after-preemption (*aapre* is set to *YES*). This means that if a Altair Grid Engine job that consumes a LO-license resource gets preempted, then this will automatically cause preemption of the corresponding LO-license request. The license will be freed and is then available for other jobs.

Manual preemption triggered in one Altair Grid Engine cluster does not provide a guarantee that the specified preemption consumer (or even a different job within the same Altair Grid Engine cluster) will get the released resources. The decision which cluster will get the released resource depends completely on the setup of the License Orchestrator cluster. Consequently it might happen that a license resource that gets available through preemption in one cluster will be given to a job in a different cluster if the final priority of the job/cluster is higher than that of the specified preemption consumer.

COMMON USE CASES

A) License consumable (without LO)

Scenario: There is a license-consumable defined that has a maximum capacity and multiple jobs compete for those by requesting one or multiple of those licenses.

Complex definition:

```
$ qconf -sc
...
license  lic  INT  <=  YES  YES  0  0  YES
...
```

The last *YES* defines the value of *aapre*. This means that the license resource will be available after preemption.

License capacity is defined on global level:

```
$ qconf -se global
...
complex_values   license=2
```

When now two jobs are submitted into the cluster then both licenses can be consumed by the jobs.

```
$ qsub -l lic=1 -b y -l h_rt=1:00:00 sleep 3600
$ qsub -l lic=1 -b y -l h_rt=1:00:00 sleep 3600
...

$ qstat -F lic
...
all.q@rgbtest                  BIPC  0/1/60   lx-amd64
       gc:license=0
3000000005 0.55476 sleep      user         r
----------------------------------------------------------------------------------
all.q@waikiki                  BIPC  0/1/10         0.00     lx-amd64
       gc:license=0
3000000004 0.55476 sleep      user         r     04/02/2015 12:32:54     1
```

Submission of a higher priority job requesting 2 licenses and resource reservation:

```
$ qsub -p 100 -R y -l lic=2 -b y -l h_rt=1:00:00 sleep 3600
```

The high priority job stays pending, it will get a reservation, but only after both lower priority jobs are expected to finish:

```
$ qstat -j 3000000006
...
reservation       1:    from 04/02/2015 13:33:54 to 04/02/2015 14:34:54
                        all.q@hookipa: 1
```

We want the high priority job to get started immediately, therefore we trigger a manual preemption of the two lower priority jobs:

```
$ qmod -p 3000000006 3000000004 3000000005 P
Accepted preemption request for preemptor candidate 3000000006
```

The lower priority jobs get preempted, the high priority job can start:

```
$ qstat
job-ID     prior   name  user state submit/start at     queue     jclass slots ja-task-ID
-----------------------------------------------------------------------------------------
3000000006 0.60361 sleep joga r 04/02/2015 12:37:50 all.q@waikiki        1
3000000004 0.55476 sleep joga P 04/02/2015 12:32:54                      1
3000000005 0.55476 sleep joga P 04/02/2015 12:32:54                      1
```

Resources which have been preempted are shown in qstat -j . In order for the preemptees to be able to resume work as soon as possible, preempted jobs get a resource reservation for the resources they released, e.g.

```
$ qstat -j 3000000004
...
preempted   1: license, slots
usage       1: wallclock=00:04:45, cpu=00:00:00, mem=0.00015 GBs, io=0.00009,
               vmem=19.414M, maxvmem=19.414M
reservation 1: from 04/02/2015 13:38:50 to 05/09/2151 19:07:05
               all.q@waikiki: 1
```

B) License managed via LO that is connected to two different Altair Grid Engine clusters

Scenario: There is a license-consumable defined that has a maximum capacity and multiple jobs from two different connected Altair Grid Engine clusters (named A and B) compete for those by requesting one or multiple of those licenses.

TODO

SEE ALSO

sge_intro(1)

COPYRIGHT

sge_share_mon

NAME

sge_share_mon - Shows status of Altair Grid Engine share tree.

SYNTAX

sge_share_mon [-a] [-c count] [-d delimiter] [-f field[,field…]] [-h] [-i interval] [-l delimiter] [-m output_mode] [-n] [-o output_file] [-r delimiter] [-s string_format] [-t] [-u] [-x]

DESCRIPTION

sge_share_mon shows the current status of the Altair Grid Engine share tree nodes. Without any options *sge_share_mon* will display a formatted view of the share tree every 15 seconds. The share tree is the configuration object for the Altair Grid Engine share tree scheduler policy.

OPTIONS

-a

Display alternate format containing usage fields defined in the *usage_weight_list* attribute of the scheduler configuration.

-c count

The number of times to collect share tree information (default is infinite).

-d delimiter

The column delimiter to use between fields (default is <TAB>).

-f field[,field…]

List of fields to print. This overrides the list of fields in the OUTPUT FORMATS section.

-h

Print a header containing the field names.

-i interval

The collection interval in seconds (default is 15).

-l

The line delimiter to use between nodes (default is <CR>).

-m output_mode

The output file fopen mode (default is “w”).

-n

Use name=value format for all fields.

-o output_file

Write output to file.

-r delimiter

The delimiter to use between collections of the share tree (default is <CR>).

-s string_format

Format of displayed strings (default is %s).

-t

Show formatted times.

-u

Show decayed usage (since timestamp) in nodes.

-x

Exclude non-leaf nodes.

OUTPUT FORMATS

For each node in the share tree, one line is printed. The output consists of:

- curr_time - the time stamp of the last status collection for this node
- usage_time - the time stamp of the last time the usage was updated
- node_name - the name of the node
- user_name - the name of the user if this is a user node
- project_name - the name of the project if this is a project node
- shares - the number of shares assigned to the node
- job_count - the number of active jobs associated to this node
- level% - the share percentage of this node amongst its siblings
- total% - the overall share percentage of this node amongst all nodes
- long_target_share - the long term target share that we are trying to achieve
- short_target_share - the short term target share that we are trying to achieve in order to meet the long term target
- actual_share - the actual share that the node is receiving based on usage
- usage - the combined and decayed usage for this node

By default, each node status line also contains the following fields:

- wallclock - the accumulated and decayed wallclock time for this node
- cpu - the accumulated and decayed CPU time for this node
- mem - the accumulated and decayed memory usage for this node. This represents the amount of virtual memory used by processes multiplied by the user and system CPU time. The value is expressed in gigabyte seconds.
- io - the accumulated and decayed I/O usage for this node
- ltwallclock - the total accumulated wallclock time for this node
- ltcpu - the total accumulated CPU time for this node
- ltmem - the total accumulated memory usage (in gigabyte seconds) for this node
- ltio - the total accumulated I/O usage for this node

If the -a option is supplied, an alternate format is displayed where the fields above following the usage field are not displayed. Instead, each node status line contains a field for each usage value defined in the usage_weight_list attribute of the scheduler configuration. The usage fields are displayed in the order that they appear in the usage_weight_list. Below are some of the supported fields.

- memvmm - the accumulated and decayed memory usage for this node. This represents the amount of virtual memory used by all processes multiplied by the wallclock run-time of the process. The value is expressed in gigabyte seconds.

- memrss - the accumulated and decayed memory usage for this node. This represents the resident set size (RSS) used by all processes multiplied by the wallclock run-time of the process. The value is expressed in gigabyte seconds. The resident set size is the amount of physical private memory plus the amount of physical shared memory being used by the process.
- mempss - the accumulated and decayed memory usage for this node. This represents the proportional set size (PSS) used by all processes multiplied by the wallclock run-time of the process. The value is expressed in gigabyte seconds. The proportional set size is the amount of physical private memory plus a proportion of the shared memory being used by the process.

NOTE

Sharetree usage is updated by the scheduler thread in every scheduling interval. When sge_share_mon is started / printing data shortly after the sharetree has been modified and before scheduler thread could updated sharetree usage data sge_share_mon will print a warning “sharetree has been modified on <date / time> and usage not yet been updated by scheduler” to stderr. Until the sharetree has been updated the per node job_count will be 0.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell uses (in the order of precedence): The name of the cell specified in the environment variable SGE_CELL, if it is set. The name of the default cell, i.e. default.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which is expected to listen for communication requests. Most installations will use a services map entry for the service “sge_qmaster” instead to define that port.

EXIT STATUS

0

on success

1

on error

FILES

<sge_root>/<cell>/common/act_qmaster

Altair Grid Engine master host file

SEE ALSO

COPYRIGHT

sge_types

NAME

sge_types - Altair Grid Engine type descriptions

DESCRIPTION

The Altair Grid Engine user interface consists of several programs and files. Some command-line switches and several file attributes are types. The syntax for these types is explained in this page.

OBJECT TYPES

These types are used for defining Altair Grid Engine configuration:

calendar_name

A calendar name is the name of a Altair Grid Engine calendar described in *sge_calendar_conf*(5).

calendar_name := *object_name*

ckpt_name

A “ckpt_name” is the name of a Altair Grid Engine checkpointing interface described in *sge_checkpoint*(5).

ckpt_name := *object_name*

complex_name

A complex name is the name of a Altair Grid Engine resource attribute described in *sge_complex*(5).

complex_name := *object_name*

host_identifier

A host identifier can be either a host name or a host group name.

host_identifier := *host_name* | *hostgroup_name*

hostgroup_name

A host group name is the name of a Altair Grid Engine host group described in *sge_hostgroup*(5). Note, to allow host group names easily be differed from host names a "@" prefix is used.

hostgroup_name := @*object_name*

host_name

A host name is the official name of a host node. Host names with a domain specification such as "gridmaster.sun.com" are called fully-qualified host names, whereas host names like "gridmaster" are called short host names. Note, there are the install time parameters default_domain and ignore_fqdn (see *sge_bootstrap*(5)) which affect how Altair Grid Engine deals with host names in general.

jc

The job class name specification is a placeholder for the name of a job class or a job class variant. When the variant part is omitted then automatically the default variant is specified.

jc := jc_name ["." variant_name]

jc_list

The job class list specification allows to reference multiple job class variants with one command.

jc_list := jc [, jc , . . .]

jsv_url

The **jsv_url** has following format:

jsv_url := *jsv_client_url* | *jsv_server_url*

jsv_server_url := [*type* ':'] [*user* **'@'**] *path*

jsv_client_url := [*type* ':'] *path*

type := **'script'**

In the moment only the type script is allowed. This means that path is either the path to a script or to a binary application which will be used to instantiate a JSV process. The type is optional till other types are supported by Altair Grid Engine.

Specifying a user is only allowed for server JSV's. Client JSV's will automatically be started as submit user and server JSV's as admin user if not other specified.

The path has always to be the absolute path to a binary or application.

memory_specifier

Memory specifiers are positive decimal, hexadecimal or octal integer constants which may be followed by a multiplier letter. Valid multiplier letters are k, K, m, M, g, G, t, T, p, P, e, E, z, Z, y and Y. Lower case letters represent a multiplier of a value 1000ˆx and capital letters a multiplier 1024ˆx where x is 1 for k/K, 2 for m/M, 3 for g/G, 4 for t/T, 5 for p/P 6 for e/E, 7 for z/Z and 8 for y/Y. If no multiplier is present, the value is just counted in bytes.

pe_name

A "pe_name" is the name of a Altair Grid Engine parallel environment described in *sge_pe*(5).

pe_name := *object_name*

project_name

A project name is the name of a Altair Grid Engine project described in *sge_project*(5).

project_name := *object_name*

queue_name

A queue name is the name of a Altair Grid Engine queue described in *sge_queue_conf*(5).

queue_name := *object_name*

tid

Task ID which identifies a task of a parallel job. Number greater or equal 0.

tid_range

A range specification of *tid*'s of the form

tid_range := *n* ['-' [*m*] [':'*s*]]

where *n* and *m* specify the lower and upper *tid* of a range of IDs. *m* has to be greater or equal *n*. If *m* is omitted this represents infinity. *s* specifies an optional step size. Default for *s* is 1 if omitted. An expression like '1-9:2' represents all uneven numbers from 1 to 9 including. '2-:2' represents all even numbers beginning from 2 to infinity.

tid_range_list

Comma separated list of *tid_range*s of the form

tid_range_list := *tid_range* [, *tid_range*, ...] | 'master' | 'slaves'.

‘1-5:2,8,10,12’ and ‘1-5:2,8-12:2’ represent the task IDs ‘1,3,5,8,10,12’. All three expressions are valid *tid_range_lists*.

The predefined keywords *master* can be used as a synonym for *‘0’* and *slaves* as equivalent for *‘1-’* if it is the only *tid_range* in one *tid_range_list*.

tid_range_lists are usually used in combination with the *-petask* switch to express resource requests or other requirements for tasks of parallel jobs.

time_specifier

A time specifier either consists of a positive decimal, hexadecimal or octal integer constant, in which case the value is interpreted to be in seconds, or is built by 3 decimal integer numbers separated by colon signs where the first number counts the hours, the second the minutes and the third the seconds. If a number would be zero it can be left out but the separating colon must remain (e.g. 1:0:1 = 1::1 means 1 hours and 1 second).

user_name

A user name can be the name of a *login*(1) user or of the Altair Grid Engine user object described in *sge_user*(5).

user_name := *object_name*

userset_name

A user set name is the name of an Altair Grid Engine access list or department described in *sge_access_list*(5).

userset_name := *object_name*

object_name

An object name is a sequence of up to 512 ASCII string characters except “\n”, “\t”, “\r”, " “,”/“,”:“,”“,”\“,”[“,”]“,”{“,”}“,”|“,”(“,”)“,”@“,”%“,”,“ or the " character itself. The sequence is also not allowed to begin with “.” or “#”.

MATCHING TYPES

These types are used for matching Altair Grid Engine configuration:

expression

A wildcard expression is a regular boolean expression that consists of one or more patterns joined by boolean operators. When a wildcard expression is used, the following definition applies:

expression = [“!”] [“(”] valExp [“)”] [AND_OR expression]* valExp = pattern | expression AND_OR = “&” | “|”

where:

“!” not operator – negate the following pattern or expression “&” and operator – logically and with the following expression “|” or operator – logically or with the following expression “(” open bracket – begin an inner expression. “)” close bracket – end an inner expression. “pattern” see the pattern definition that’s follow

The expression itself should be put inside quotes (“”) to ensure that clients receive the complete expression.

e.g.

```
"(lx*|sol*)&*64*" any string beginning with either "lx" or
                  "sol" and containing "64"
"rh_3*&!rh_3.1"   any string beginning with "rh_3", except
                  "rh_3.1"
```

xor_expression

A XOR expression is a special type of expression, that can only be used with RSMAPs (see sge_resource_map(5)). Multiple expressions can be combined to a XOR expression with the XOR operator:

xor_expression = [expression] XOR [expression]

where:

“XOR” xor operator – either “^” or “^^”

pattern

When patterns are used the following definitions apply:

“*” matches any character and any number of characters (between 0 and inv). "?" matches any character. It cannot be no character "." is the character ".". It has no other meaning "\" escape character. "\\" = "\", "\" = "*“,”\?" = “?” “[. . .]” specifies an array or a range of allowed characters for one character at a specific position. Character ranges may be specified using the a-z notation. The caret symbol (^) is not interpreted as a logical not; it is interpreted literally.

For more details please see *fnmatch*(5)

The pattern itself should be put inside quotes (“”) to ensure that clients receive the complete pattern.

range

The task range specifier has the form

n[-m[:s]][,n[-m[:s]], ...] or n[-m[:s]][n[-m[:s]] ...]

and thus consists of a comma or blank separated list of range specifiers n[-m[:s]]. The ranges are concatenated to the complete task id range. Each range may be a single number, a simple range of the form n-m or a range with a step size.

soft_range

The range specifier for -soft requested consumables has the form

n-m[:s][|n-m[:s]| ...] (n < m) (n,m,s > 0)

and thus consists of a comma separated list of range specifiers n-m[:s] with optional step argument. Each range may be a simple range of the form "n-m" or a range with a step size "n-m:s". The value for n must be less than the value for m. The values of n, m and s must be larger than zero. If the type of the consumable complex is MEMORY or DOUBLE the n,m and s specifiers are positive rational numbers. For INT and RSMAP n,m and s must be a positive integer number.

wc_ar

The wildcard advance reservation (AR) specification is a placeholder for AR ids, AR names including AR name patterns. An AR id always references one AR, while the name and pattern might reference multiple ARs.

wc_ar := ar_id | ar_name | pattern

wc_ar_list

The wildcard advance reservation (AR) list specification allows to reference multiple ARs with one command.

wc_ar_list := wc_ar [, wc_ar , ...]

wc_host

A wildcard host specification (wc_host) is a wildcard expression which might match one or more hosts used in the cluster. The first character of that string never begins with an at-character ('@'), even if the expression begins with a wildcard character.

e.g.

```
*    all hosts
a*   all host beginning with an 'a'
```

wc_hostgroup

A wildcard hostgroup specification (wc_hostgroup) is a wildcard expression which might match one or more hostgroups. The first character of that string is always an at-character ('@').

More information concerning hostgroups can be found in *sge_hostgroup*(5)

e.g.

```
@*          all hostgroups in the cluster
@solaris    the @solaris hostgroup
```

wc_job

The wildcard job specification is a placeholder for job ids, job names including job name patterns. A job id always references one job, while the name and pattern might reference multiple jobs.

wc_job := job-id | job-name | pattern

wc_job_task

The wildcard job specification is a placeholder for job ids (with optional task range), job names including job name patterns. A job id always references one job (or optionally multiple tasks of the job), while the name and pattern might reference multiple jobs.

wc_job_task := job-id [. range] | job-name | pattern

wc_job_range

The wildcard job range specification allows to reference specific array tasks for one or multiple jobs. The job is referenced via wc_job and in addition gets a range specifier for the array tasks.

wc_job_range := wc_job [-t range]

wc_job_list

The wildcard job list specification allows to reference multiple jobs with one command.

wc_job_list := wc_job [, wc_job , . . .]

wc_job_task_list

The wildcard job list specification allows to reference multiple jobs (or tasks) with one command.

wc_job_task_list := wc_task_job [[,] wc_task_job [,] . . .]

wc_job_range_list

The wildcard job range list (wc_job_range_list) is specified by one of the following forms:

```
wc_job[ -t range][{, }wc_job[ -t range]{, }...]
```

If present, the task_range restricts the effect of the qmod operation to the array job task range specified as suffix to the job id (see the -t option to *qsub*(1) for further details on array jobs).

wc_qdomain

wc_qdomain := *wc_cqueue* "@" *wc_hostgroup*

A wildcard expression queue domain specification (wc_qdomain) starts with a wildcard expression cluster queue name (wc_cqueue) followed by an at-character '@' and a wildcard expression hostgroup specification (wc_hostgroup).

wc_qdomain are used to address a group of queue instances. All queue instances residing on a hosts which is part of matching hostgroups will be addressed. Please note, that wc_hostgroup always begins with an at-character.

e.g.

```
*@@*         all queue instances whose underlying
             host is part of at least one hostgroup
a*@@e*       all queue instances begins with a whose underlying
             host is part of at least one hostgroup begin with e
*@@solaris   all queue instances on hosts part of
             the @solaris hostgroup
```

wc_cqueue

A wildcard expression cluster queue specification (wc_cqueue) is a wildcard expression which might match one or more cluster queues used in the cluster. That string never contains an at-character ('@'), even if the expression begins with a wildcard character.

e.g.

```
*          all cluster queues
a*         all cluster queues beginning with an 'a'
a*&!adam   all cluster queues beginning with an 'a',but not adam
```

wc_qinstance

wc_qinstance := wc_cqueue "@" wc_host

A wildcard expression queue instance specification (wc_qinstance) starts with a wildcard expression cluster queue name (wc_cqueue) followed by an at-character '@' and a wildcard expression hostname (wc_host).

wc_qinstance expressions are used to address a group of queue instances whose underlying hostname matches the given expression. Please note that the first character of wc_host does never match the at-character '@'.

e.g.

```
*@*       all queue instances in the cluster
*@b*      all queue instances whose
          hostname begins with a 'b'
*@b*|c*   all queue instances whose
          hostname begins with a 'b' or 'c'
```

wc_queue

wc_queue := wc_cqueue | wc_qdomain | wc_qinstance

A wildcard queue expression (wc_queue) might either be a wildcard expression cluster queue specification (wc_cqueue) or a wildcard expression queue domain specification (wc_qdomain) or a wildcard expression queue instance specification (wc_qinstance).

e.g.

```
big_*1      cluster queues which begin with
            "big_" and end with "1"
big_*&!*1   cluster queues which begin with
            "big_" ,but does not end with "1"
*@fangorn   all qinstances residing on host
            fangorn
```

wc_queue_list

wc_queue_list := wc_queue [“,” wc_queue “,” . . .]

Comma separated list of wc_queue elements.

e.g.

```
big, medium_*@@sol*, *@fangorn.sun.com
```

wc_user

A wildcard user name pattern is either a wildcard user name specification or a full user name.

wc_user := user_name | pattern

wc_user_list

A list of user names.

wc_user_list := wc_user [, wc_user , . . .]

wc_project

A wildcard project name pattern is either a wildcard project name specification or a full project name.

wc_project := project | pattern

wc_pe_name

A wildcard parallel environment name pattern is either a wildcard pe name specification or a full pe name.

wc_pe_name := pe_name | pattern

parallel_env n[-[m]]|[-]m,...

Parallel programming environment (PE) to select for an AR. The range descriptor behind the PE name specifies the number of parallel processes to be run. Altair Grid Engine will allocate the appropriate resources as available. The *sge_pe*(5) manual page contains information about the definition of PEs and about how to obtain a list of currently valid PEs.

You can specify a PE name which uses the wildcard character, "*". Thus the request "*pvm*" will match any parallel environment with a name starting with the string "pvm". In the case of multiple parallel environments whose names match the name string, the parallel environment with the most available slots is chosen.

The range specification is a list of range expressions of the form "n-m", where n and m are positive, non-zero integers. The form "n" is equivalent to "n-n". The form "-m" is equivalent to "1-m". The form "n-" is equivalent to "n-infinity". The range specification is processed as follows: The largest number of queues requested is checked first. If enough queues meeting the specified attribute list are available, all are reserved. If not, the next smaller number of queues is checked, and so forth.

date_time

The date_time value must conform to [[CC]]YY]MMDDhhmm[.ss], where:

e.g.

```
CC denotes the century in 2 digits.
YY denotes the year in 2 digits.
MM denotes the month in 2 digits.
```

```
DD denotes the day in 2 digits.
hh denotes the hour in 2 digits.
mm denotes the minute in 2 digits.
ss denotes the seconds in 2 digits (default 00).
```

time

The time value must conform to hh:mm:ss, or seconds where:

e.g.

```
hh denotes the hour in 2 digits.
mm denotes the minute in 2 digits.
ss denotes the seconds in 2 digits (default 00).
seconds  is a number of seconds (is used for duration values)

If any of the optional date fields are omitted, the corresponding value of the current date is as
Use of this option may cause unexpected results if the clocks of the hosts in the Altair Grid Eng
and *sge_execd*(8) are invoked.
```

name

The name may be any arbitrary alphanumeric ASCII string, but may not contain a leading digit, “\n”, “\t”, “\r”, “/”, “:”, “@”, “\”, "*“, or”?".

SEE ALSO

qacct(1), *qconf*(1), *qquota*(1), *qsub*(1), *qrsub*(1)

COPYRIGHT

sgepasswd

NAME

sgepasswd - Modify the Altair Grid Engine password file

SYNTAX

sgepasswd [[**-D domain**] **-d user**]

sgepasswd [**-D domain**] [**user**]

DESCRIPTION

sgepasswd modifies the Altair Grid Engine password file *sgepasswd*(5).

This file contains a list of usernames and their windows password in encrypted form. *sge_execd*(8) and *sge_shepherd*(8) on hosts running Microsoft Windows as operating systems use this information to start jobs for a certain user.

Each user can use the *sgepasswd* application to add a new entry for the own user account. It is also possible to change the stored password with *sgepasswd* as far as the user knows the old one.

The root user additionally has the permission to change the password entries for other user accounts. Root can also delete existing entries.

The *sgepasswd* application is only available on non-Windows hosts. In order to be able to modify the *sgepasswd*(5) file, the *sgepasswd* application needs to run with the right user ID set (usually the one of root). To achieve this, the setuid bit has to be set in the file permissions and the *sgepasswd* application has to be owned by the right user (usually root). The *sge_qmaster*(8) installation script sets the permissions and owner properly if it is requested.

The “ls -l” output of the *sgepasswd* application should look like this:

-r-sr-xr-x 1 root root 4136176 Dec 10 15:51 sgepasswd

However, the setuid mechanism does not work if the *sgepasswd* application resides on a file system that is mounted with the “nosuid” option. If policies require the $SGE_ROOT directory to be located on such a file system, copying the *sgepasswd* application to a different location is a possible workaround for this problem.

OPTIONS

-D domain

Per default *sgepasswd* will add/modify the current Unix username without domain specification. This switch can be used to add a domain specification in front of the current user name. Consult your Microsoft Windows documentation to get more information about domain users.

-c

Only root can use this parameter to convert an existing RC4 encrypted sgepasswd into a AES_256_CBC encrypted *sgepasswd*(5) file.

-d user

Only root can use this parameter to delete entries from the *sgepasswd*(5) file for other users but normal users can delete their own user entry from the *sgepasswd*(5) file.

-l

List all user names in *sgepasswd*(5) file.

-n oldkeyfile

Only root can use this parameter to re-encrypt an existing sgepasswd file encrypted with oldkeyfile with the current keyfile. If during upgrade the CA is regenerated the old keyfiles are saved under /var/sgeCA/port${SGE_QMASTER_PORT}/${SGE_CELL}.backup and $SGE_ROOT/$SGE_CELL/common/sgeCA.backup In this scenario oldkeyfile would be: /var/sgeCA/port${SGE_QMASTER_PORT}/${SGE_CELL}.backup/private/key.pem Don't apply the -n option multiple times.

-help

Prints a listing of all options.

-version

Display version information for the *sgepasswd* command.

ENVIRONMENTAL VARIABLES

SGE_CERTFILE

Specifies the location of public key file. Per default *sgepasswd* will use the file $SGE_ROOT/$SGE_CELL/common/s

SGE_KEYFILE

If set, specifies the location of the private key file. Default is
/var/sgeCA/port${SGE_QMASTER_PORT}/${SGE_CELL}/private/key.pem

SGE_RANDFILE

If set, specifies the location of the seed used to create encrypted versions of user passwords.
Default is /var/sgeCA/port${SGE_QMASTER_PORT}/${SGE_CELL}/private/rand.seed

SEE ALSO

sge_intro(1), *sgepasswd*(5)

COPYRIGHT

submit

NAME

qsub - submit a batch job to Altair Grid Engine.

qsh - submit an interactive X-windows session to Altair Grid Engine.

qlogin - submit an interactive login session to Altair Grid Engine.

qrsh - submit an interactive rsh session to Altair Grid Engine.

qalter - modify a pending or running batch job in Altair Grid Engine.

qresub - submit a copy of an existing Altair Grid Engine job.

SYNTAX

qsub [**options**] [**command** [**command_args**] | -- [**command_args**]]

qsh [**options**] [**-- xterm_args**]

qlogin [**options**]

qrsh [**options**] [**command** [**command_args**]]

qalter [**options**] **wc_job_range_list** [**–** [**command_args**]]

qalter [**options**] **-u user_list | -uall** [**–** [**command_args**]]

qresub [**options**] **job_id_list**

DESCRIPTION

The *qsub* command submits batch jobs to the Altair Grid Engine queuing system. Altair Grid Engine supports single- and multiple-node jobs. **Command** can be a path to a binary or a script (see **-b** below) which contains the commands to be run by the job using a shell (for example, *sh*(1) or *csh*(1)). Arguments to the command are given as **command_args** to *qsub*. If **command** is handled as a script, it is possible to embed flags in the script. If the first two characters of a script line either match '#$' or are equal to the prefix string defined with the **-C** option described below, the line is parsed for embedded command flags.

The *qsh* command submits an interactive X-windows session to Altair Grid Engine. An *xterm*(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the *-display qsh* option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by *qsh* that appropriate resources to execute the job are not

available. **xterm_args** are passed to the *xterm*(1) executable. Note, however, that the *-e* and *-ls* xterm options do not work with *qsh .*

The *qlogin* command is similar to *qsh* in that it submits an interactive job to the queuing system. It does not open an *xterm*(1) window on the X display, but uses the current terminal for user I/O. Usually, *qlogin* establishes a built-in connection with the remote host using Altair Grid Engine built-in data transport mechanisms, but can also be configured to establish a connection with the remote host using an external mechanism like *ssh*(1)/*sshd*(8).

These commands can be configured with the **qlogin_daemon** (server-side, *built-in* by default, otherwise something like /usr/sbin/sshd) and **qlogin_command** (client-side, *built-in* by default, otherwise something like /usr/bin/ssh) parameters in the global and local configuration settings of *sge_conf*(5). The client-side command is automatically parameterized with the remote host name and port number to which to connect, resulting in an invocation like

```
/usr/bin/ssh my_exec_host 2442
```

for example, which is not a format *ssh*(1) accepts, so a wrapper script must be configured instead:

```
#!/bin/sh
HOST=$1
PORT=$2
/usr/bin/ssh -p $PORT $HOST
```

The *qlogin* command is invoked exactly like *qsh* and its jobs can only run on INTERACTIVE queues. *qlogin* jobs can only be used if the *sge_execd*(8) is running under the root account.

The *qrsh* command is similar to *qlogin* in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, *qrsh* establishes a *built-in* connection with the remote host. If no command is given to *qrsh*, an interactive session is established. It inherits all SGE_ environment variables plus SHELL, HOME, TERM, LOGNAME, TZ, HZ, PATH and LANG. The server-side commands used can be configured with the **rsh_daemon** and **rlogin_daemon** parameters in the global and local configuration settings of *sge_conf*(5). *Built-in* interactive job support is used is used if the parameters are not set. If the parameters are set, they should be set to something like /usr/sbin/sshd. On the client side, the **rsh_command** and **rlogin_command** parameters can be set in the global and local configuration settings of *sge_conf*(5). Use the cluster configuration parameters to integrate mechanisms like *ssh* supplied with the operating system. In order to use *ssh*(1)/*sshd*(8), configure for both the **rsh_daemon** and the **rlogin_daemon** "/usr/sbin/sshd -i" and for the **rsh_command** or **rlogin_command** "/usr/bin/ssh".

qrsh jobs can only run in INTERACTIVE queues unless the option **-now no** is used (see below). They can also only be run if the *sge_execd*(8) is running under the root account.

qrsh provides an additional useful feature for integrating with interactive tools providing a specific command shell. If the environment variable **QRSH_WRAPPER** is set when *qrsh* is invoked, the command interpreter pointed to by **QRSH_WRAPPER** will be executed to run *qrsh* commands instead of the user's login shell or any shell specified in the *qrsh* commandline. The options **-cwd** and **-display** only apply to batch jobs.

qalter can be used to change the attributes of pending jobs. For array jobs with a mix of running and pending tasks (see the **-t** option below), modification with *qalter* only affects the pending tasks. *Qalter* can change most of the characteristics of a job (see the corresponding statements in the OPTIONS section below), including those which were defined as embedded flags in the script file (see above). Some submit options, such as the job script, cannot be changed with *qalter.*

qresub allows the user to create jobs as copies of existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they were copied, except with a new job ID and with a cleared hold state. The only modification to the copied jobs supported by *qresub* is assignment of a new hold state with the **-h** option. This option can be used to first copy a job and then change its attributes via *qalter*.

Only a manager can use *qresub* on jobs submitted by another user. Regular users can only use *qresub* on their own jobs.

For *qsub*, *qsh*, *qrsh*, and *qlogin* the administrator and the user may define default request files (see sge_request*(5)*) which can contain any of the options described below. If an option in a default request file is understood by *qsub* and *qlogin* but not by *qsh* the option is silently ignored if *qsh* is invoked. Thus you can maintain shared default request files for both *qsub* and *qsh*.

A cluster-wide default request file may be placed under $SGE_ROOT/$SGE_CELL/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags, and the options in the default request files are processed in the following order:

```
left to right in the script line,
left to right in the default request files,
from top to bottom in the script file (_qsub_ only),
from top to bottom in default request files,
from left to right in the command line.
```

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note: the *-clear* option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with *qalter*.

The options described below can be requested as either hard or soft. By default, all requests are considered hard until the **-soft** option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

OPTIONS

-@ optionfile

Forces *qsub*, *qrsh*, *qsh*, or *qlogin* to use the options contained in **optionfile**. The indicated file may contain all valid options. Comment lines must start with a "#" sign.

-a date_time

Available for *qsub* and *qalter* only.

Defines or redefines the time and date at which a job is eligible for execution. **Date_time** conforms to [[CC]YY]MMDDhhmm[.SS]; for details, please see **date_time** in *sge_types*(1).

If this option is used with *qsub* or if a corresponding value is specified in *qmon*, a parameter named **a** with the format CCYYMMDDhhmm.SS will be passed to the defined JSV instances (see the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5)).

-ac variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Adds the given name/value pair(s) to the job's context. **Value** may be omitted. Altair Grid Engine appends the given argument to the list of context variables for the job. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important here. The variable name must not start with the characters "+", "-", or "=".

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) *QALTER* allows changing this option even while the job executes.

-adds parameter key value

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to add additional entries to list-based job parameters including resource requests, job context, environment variables and more. The **-mods** and **-clears** switches can be used to modify or remove a single entry of a job parameter list.

The parameter argument specifies the job parameter that should be enhanced. The names used here are names of command-line switches that are also used in job classes or JSV to address job parameters. Currently the **-adds** switch supports the following parameters: **ac**, **CMDARG**, **cwd**, **e**, **hold_jid**, **i**, **l_hard**, **l_soft**, **M**, **masterl**, **masterq**, **o**, **q_hard**, **q_hard**, **rou**, **S** and **v**. The same set of parameters is also supported by the **-mods** and **-clears** switches. The **-clearp** switch allows resetting all list-based parameters mentioned above as well as

non-list-based parameters. Find corresponding non-list-based parameter names in the **-clearp** section below.

Please note that the **cwd** parameter is a list-based parameter that can be addressed with the **-adds**, **-mods** and **-clears** switches although this list can only have one entry.

The key argument depends on the used parameter argument. For the **ac** and **v** parameter it has to specify the name of a variable that should be added to either the job context or environment variable list. For the parameters **o**, **i**, **e** or **S** it is a hostname. An empty key parameter might be used to define a default value that is not host-specific. The key of **l_hard** or **l_soft** has to refer to a resource name (name of a complex entry) whereas **q_hard**, **q_soft** and **masterq** expect a queue name. **CMDARG** expects a string that should be passed as a command-line argument, **hold_jid** a name or job ID of a job and **M** a mail address.

All parameter/key combinations expect a value argument. For the **CMDARG**, **q_hard**, **q_soft**, **hold_jid**, **M**, and **rou** parameters, this value has to be an empty argument. **ac**, **v**, **l_hard**, and **l_soft** also allow empty values.

Independent of the position within the command line, the switches **-adds**, **-mods**, and **-clears** will be evaluated after modifications of all other switches that are passed to the command used to submit the job, or *qalter*, and the sequence in which they are applied is the same as specified on the command line.

If the **-adds** parameter is used to change a list-based job parameter that was derived from a job class, this operation might be rejected by the Altair Grid Engine system. This can happen if within the job class access specifiers were used that do not allow adding new elements to the list. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

-ar ar_id|ar_name

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

Assigns the submitted job to be a part of an existing Advance Reservation. The complete list of existing Advance Reservations can be obtained using the *qrstat*(1) command.

Note that the **-ar** option implicitly adds the **-w e** option if it is not otherwise requested. Also, the advance reservation name (ar_name) can be used only when qmaster_param **SUBMIT_JOBS_WITH_AR_NAME** is set to true.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

When this option is used for a job or when a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **ar**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-A account_string

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Identifies the account to which the resource consumption of the job should be charged. The **account_string** should conform to the **name** definition in *sge_types*(1). In the absence of

this parameter Altair Grid Engine will place the default account string "sge" in the accounting record of the job.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value is passed to defined JSV instances as a parameter with the name **A**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-bgio bgio_params

This option will bypass the problem that the controlling terminal will suspend the qrsh process when it is reading from STDIN or writing to STDOUT/STDERR file descriptors.

Available for *qrsh* with built-in interactive job support mechanism only.

Supported if e.g. "&" is used to start qrsh in the background of a terminal. The terminal must support job control and must also have a supported tty assigned.

Supported bgio_params options are **nr** (no read), **fw** (forced write) and **bw=<size>** (buffered write up to the specified buffer size). Options can be combined by using the "," character as a delimiter (no spaces allowed):

bgio_params nr|bw=<size>|fw[,nr|bw=<size>|fw,...]

nr: no read
If the user terminal supports job control, qrsh will not read from STDIN when it is running in the background. If a user is entering input at the terminal the default behavior often is that the process running in the background is suspended when it reads the user input from STDIN. This is done by the user's terminal for all background jobs which try to read from STDIN. When using the "nr" option, qrsh will not read from STDIN as long it is running in the background.

fw: force write
If the "stty tostop" option is active for the user's terminal any job running in the background of the terminal will be suspended when it tries to write to STDOUT or STDERR. The "fw" option is used to tell qrsh to ignore this setting and force writing without being suspended.

bw=<size>: buffered write
If the user terminal has the "stty tostop" option set (background jobs will be suspended when writing to STDOUT or STDERR) it is possible to simply buffer the messages in the qrsh client to avoid being suspended by using this option. qrsh will write to STDOUT or STDERR if one of the following occurs:

- When the process is in the foreground again
- When the buffer is full
- When qrsh is terminating

-binding [binding_instance] binding_strategy

Available for *qsub*, *qalter*, *qrsh*, *qsh*, and *qlogin* only.

A job can request a specific processor core binding (processor affinity) by using this parameter. This request is treated since version 8.1 as a hard resource request, i.e. the job is only dispatched to a host which is able to fulfill the request. In contrast to previous versions the request is now processed in the Altair Grid Engine scheduler component.

To force Altair Grid Engine to select a specific hardware architecture, please use the **-l** switch in combination with the complex attribute **m_topology**.

binding_instance is an optional parameter. It might be any of **env**, **pe**, or **set**, depending on which instance should accomplish the job-to-core binding. If the value for **binding_instance** is not specified, **set** will be used.

env means that only the environment variable **SGE_BINDING** will be exported to the environment of the job. This variable contains the selected operating system internal processor numbers. They might be more than selected cores in presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated. This variable is also available for real core binding when **set** or **pe** was requested.

pe means that the information about the selected cores appears in the fourth column of the **pe_hostfile**. Here the logical core and socket numbers are printed (they start at 0 and have no holes) in colon-separated pairs (i.e. 0,0:1,0 which means core 0 on socket 0 and core 0 on socket 1). For more information about the $pe_hostfile check sge_pe*(5)*

set (default if nothing else is specified). The binding strategy is applied by Altair Grid Engine. How this is achieved depends on the underlying operating system of the execution host where the submitted job will be started.

On Solaris 10 hosts, a processor set will be created in which the job can run exclusively. Because of operating system limitations at least one core must remain unbound. This resource could of course be used by an unbound job.

On Linux (lx-amd64 or lx-x86) hosts a processor affinity mask will be set to restrict the job to run exclusively on the selected cores. The operating system allows other unbound processes to use these cores. Please note that on Linux the binding requires a Linux kernel version of 2.6.16 or greater. It might even be possible to use a kernel with a lower version number but in that case additional kernel patches would have to be applied. The **loadcheck** tool in the utilbin directory can be used to check the host's capabilities. You can also use **-sep** in combination with **-cb** of the *qconf*(1) command to identify whether Altair Grid Engine is able to recognize the hardware topology.

PE-jobs and core-binding: As of version 8.6 the behavior of the given amount of cores to bind changed to mean the amount of cores per PE-task. A sequential job (without **-pe** specified) counts as a single task. Prior to version 8.6 the **<amount>** was per host, independent of the number of tasks on that host. Example: *"qsub -pe mype 7-9 -binding linear:2"* since version 8.6 means that on each host 2 cores are going to be bound for each task that is scheduled on it (the total number of tasks and possibly of hosts is unknown at submit-time due to the given range). This enables a fast way of deploying hybrid parallel jobs, meaning distributed jobs that are also multi-threaded (e.g. MPI + OpenMP).

Possible values for **binding_strategy** are as follows:

```
linear:<amount>[:<socket>,<core>]
linear_per_task:<amount>
```

```
striding:<amount>:<n>[:<socket>,<core>]
striding_per_task:<amount>:<n>
explicit:[<socket>,<core>:...]<socket>,<core>
explicit_per_task:[<socket>,<core>:...]<socket>,<core>
balance_sockets:<amount>
pack_sockets:<amount>
one_socket_balanced:<amount>
one_socket_per_task:<amount>
```

For the binding strategies linear and striding, there is an optional socket and core pair attached. These denote the mandatory starting point for the first core to bind on. For linear_per_task, striding_per_task, balance_sockets, pack_sockets, one_socket_balanced, and one_socket_per_task, this starting point cannot be specified. All strategies that are not suffixed with **_per_task** mean per host, i.e. all tasks taken together on each host have to adhere to the binding strategy. All strategies with the suffix **_per_task** mean per task, i.e. the tasks have to follow the strategy individually - independent of all other tasks. This is less strict and usually more tasks can fit on a host. For example when using **linear**, all tasks on a host have to be "next to each other", while when using **linear_per_task**, there can be gaps between the tasks, but all cores of each task have to be "next to each other". More details below.

In the following section a number of examples are given. The notation for these examples is as follows: An empty example host has 8 slots and 8 cores. The core topology is displayed here as "SCCCC SCCCC", where "S" is a socket, "C" is a free core, and a small "c" denotes an already-occupied core.

linear / **linear_per_task** means that Altair Grid Engine tries to bind the job on **amount** successive cores per task. **linear** is a "per host"-strategy, meaning that all cores for all tasks together have to be linear on a given host. **linear_per_task** is less strict and only requires that all cores within a task have to be linear.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding linear:2
(Host) Scccc SccCC (tasks: 3)

On two hosts with already occupied cores:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear:2 would lead to:
(Host 1) Scccc SCcCC (tasks: 2)
(Host 2) SCccc Scccc (tasks: 3)
5 tasks are scheduled and all cores of these tasks are linear.

On the other hand, if one uses linear_per_task, one would get:
(Host 1) SCCCC SCcCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding linear_per_task:2 would lead to:
(Host 1) Scccc SCccc (tasks: 3)
```

```
(Host 2) SCccc Scccc (tasks: 3)
leading to a total of 6 tasks, each task having 2 linear cores.
```

striding / **striding_per_task** means that Altair Grid Engine tries to find cores with a certain offset. It will select **amount** of empty cores per task with an offset of **n** -1 cores in between. The start point for the search algorithm is socket 0 core 0. As soon as **amount** cores are found they will be used to do the job binding. If there are not enough empty cores or if the correct offset cannot be achieved, there will be no binding done.

Example:

```
(Host) SCCCC SCCCC (tasks: 0)
qsub -pe mype 1-3 -binding striding:2:2
(Host) ScCcC ScCcC (tasks: 2)
```

This is the maximum amount of tasks that can fit on this host for **striding**.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding:2 would lead to:
(Host 1) SCcCc SccCc (tasks: 2)
(Host 2) ScccC ScCcC (tasks: 2)
```

4 tasks are scheduled and the remaining free cores cannot
be used by this job, as that would violate striding per host.

On the other hand, if one uses **striding_per_task**, one would get:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 4-6 -binding striding_per_task:2
(Host 1) Scccc SccCc (tasks: 3)
(Host 2) ScccC Scccc (tasks: 3)
```

leading to a total of 6 tasks scheduled, as the striding tasks can be interleaved.

explicit / **explicit_per_task** binds the specified sockets and cores that are mentioned in the provided socket/core list. Each socket/core pair has to be specified only once. If any socket/core pair is already in use by a different job this host is skipped. Since for **explicit** no amount can be specified, one core per task is used. Therefore, using **explicit** would lead to as many tasks on each host as the number of socket/core pairs (or less). **explicit_per_task** means that each task gets as many cores as socket/core pairs are requested. It also implies that only one task per host can be scheduled, as each task has to have exactly that binding, which can only exist once on each host.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 4)
(Host 2) SCccC ScCCC (tasks: 3)
```

Each task gets one core. On host 2 there could be another task, but only up to 7 tasks were requested.

On the other hand, with **explicit_per_task**, one would get:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding explicit_per_task:0,1:0,2:1,0:1,3
(Host 1) SCccC ScCCc (tasks: 1)
(Host 2) SCccC ScCCc (tasks: 1)
```

Each task gets 4 cores.

balance_sockets binds **<amount>** free cores starting with the socket with the least cores bound by Altair Grid Engine. It iterates through all sockets, each time filling the socket with the least amount of bound cores, until no more tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3
(Host 1) ScccC ScccC (tasks: 2)
(Host 2) ScccC ScccC (tasks: 2)
```

Socket 0 has no occupied cores, so a task is placed there.
Socket 1 then has no occupied cores, but socket 0 has 3, therefore
socket 1 is chosen for the next task. Only 2 free cores remain,
which is not enough for another task.

On two hosts with already-occupied cores:

```
(Host 1) SCCCC ScCCC (tasks: 0)
(Host 2) SCcCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding balance_sockets:3 would lead to:
(Host 1) ScccC Scccc (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

pack_sockets binds **<amount>** free cores starting with the socket with the most cores already bound by Altair Grid Engine, i.e. the socket with the least free cores. It iterates through all sockets, each time filling the socket with the most amount of bound cores, until no more

tasks can be placed on that host or all requested tasks are placed. There can be as many tasks on a host as free cores divided by **<amount>**.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)_
qsub -pe mype 2-7 -binding pack_sockets:2
(Host 1) Scccc Scccc (tasks: 4)
(Host 2) Scccc SccCC (tasks: 3)
```

There is no socket with bound cores, thus the first task is placed on socket 0.
The next task is also placed on socket 0, as this is the socket with the most bound cores and it has enough free cores for another task. With this, socket 0 is full.
Socket 1 is filled in the same way, as is host 2.

On two hosts with already-occupied cores:

```
(Host 1) SccCC ScCCC (tasks: 0)
(Host 2) SCccC SCCcC (tasks: 0)
qsub -pe mype 2-7 -binding pack_sockets:2 would lead to:
(Host 1) SCCcc ScccC (tasks: 2)
(Host 2) Scccc ScccC (tasks: 2)
```

Here, first socket 0 is filled. Then socket 1. Only one core remains free on socket 1, which is not enough for a task. So host 1 is full. Host 2 is filled in the same way.

one_socket_balanced / **one_socket_per_task** binds only one socket, either per host or per task. This only works if there is at least one socket available with enough free cores. **one_socket_balanced** takes the socket with the most free cores, **one_socket_per_task** goes from left to right and takes each socket on which a task can be placed.

Example:

```
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket_balanced:2
(Host 1) Scccc SCCCC (tasks: 2)
(Host 2) Scccc SCCCC (tasks: 2)
There can only be one socket per host, and therefore 4 tasks
can be scheduled in total.

With **one\_socket\_per_task** on the other hand, one would get
(Host 1) SCCCC SCCCC (tasks: 0)
(Host 2) SCCCC SCCCC (tasks: 0)
qsub -pe mype 2-7 -binding one_socket\_per_task:2
(Host 1) SccCC SccCC (tasks: 2)
(Host 2) SccCC SccCC (tasks: 2)
Again, 4 tasks can be scheduled, but now they are distributed across
4 sockets.
```

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in qmon is specified, these values will be passed to defined JSV instances as parameters with the names **binding_strategy**, **binding_type**, **binding_amount**, **binding_step**, **binding_socket**, **binding_core**, **binding_exp_n**, **binding_exp_socket<id>**, **binding_exp_core<id>**.

Please note that the length of the socket/core value list of the explicit binding is reported as **binding_exp_n**. **<id>** will be replaced by the position of the socket/core pair within the explicit list (0 <= **id** < **binding_exp_n**). The first socket/core pair of the explicit binding will be reported with the parameter names **binding_exp_socket0** and binding_exp_core0.

The following values are allowed for **binding_strategy**: **linear_automatic**, **linear**, **striding**, **striding_automatic**, **linear_per_task**, **striding_per_task**, **explicit**, and **explicit_per_task**. The value **linear_automatic** corresponds to the command-line request -binding linear:<N>. Hence **binding_amount** must be set to the amount of requested cores. The value **linear** corresponds to the command-line request "-binding linear:<N>:<socket>,<core>". In addition to the **binding_amount**, the start socket (**binding_socket**) and start core (**binding_core**) must be set. Otherwise the request is treated as "-binding linear:<N>:0,0" which is different from "-binding linear:<N>". The same rules apply to **striding_automatic** and **striding**. In the automatic case the scheduler seeks free cores, while in the non-automatic case the scheduler starts to fill up cores at the position specified in **binding_socket** and **binding_core** if possible (otherwise it skips the host).

Values that do not apply to the specified binding will not be reported to JSV. E.g. **binding_step** will only be reported for the striding binding, and all **binding_exp_*** values will passed to JSV if explicit binding was specified.

If the binding strategy should be changed with JSV, it is important to set all parameters that do not belong to the selected binding strategy to zero, to avoid combinations that could get rejected. E.g., if a job requesting **striding** via the command line should be changed to **linear**, the JSV has to set **binding_step** and possibly **binding_exp_n** to zero, in addition to changing **binding_strategy** (see the -jsv option below or find more information concerning JSV in *sge_jsv*(5)).

If only some cores of a given host should be made available for core binding (e.g. when this host is running processes outside of Altair Grid Engine), a **load sensor** can be used (see sge_execd(8)). This **load sensor** should return as "m_topology_inuse" the topology of the host, with the cores to be masked out marked with a lower case "c".

Example:

```
A host with a topology like
examplehost: SCCCCSCCCC
should never bind its last two cores, as these are reserved for processes
outside of Altair Grid Engine.

In this case a load sensor has to be configured on this host, returning
examplehost:m_topology_inuse:SCCCSCCcc
```

-b y[es]|n[o]

Available for *qsub*, *qrsh* only. Altair Grid Engine also supports modification via *qalter*.

Gives the user the ability to indicate explicitly whether **command** should be treated as a binary or a script. If the value of **-b** is 'y', the **command** may be a binary or a script. The **command** might not be accessible from the submission host. Nothing except the path of the **command** will be transferred from the submission host to the execution host. Path aliasing will be applied to the path of **command** before **command** is executed.

If the value of **-b** is 'n', **command** needs to be a script and will be handled as script. The script file has to be accessible by the submission host. It will be transferred to the execution host. *qsub*/*qrsh* will search directive prefixes within the script. *qsub* will implicitly use **-b n** whereas *qrsh* will apply the **-b y** option if nothing else is specified.

qalter can only be used to change the job type from binary to script when a script is additionally specified with *-CMDNAME*.

Please note that submission of **command** as a script (**-b n**) can have a significant performance impact, especially for short-running jobs and big job scripts. Script submission adds a number of operations to the submission process. The job script needs to be:

```
- parsed at client side (for special comments)
- transferred from submit client to qmaster
- spooled in qmaster
- transferred to execd at job execution
- spooled in execd
- removed from spooling both in execd and qmaster once the job is done
```

If job scripts are available on the execution nodes, e.g. via NFS, binary submission can be the better choice.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **b**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-CMDNAME command

Only available in Altair Grid Engine. Available for *qalter* only.

Changes the command (script or binary) to be run by the job. In combination with the **-b** switch it is possible to change binary jobs to script jobs and vice versa.

The value specified as the command during the submission of a job will be passed to defined JSV instances as a parameter with the name **CMDNAME**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-c occasion_specifier

Available for *qsub* and *qalter* only.

Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see *sge_checkpoint*(5)) referenced by the *qsub -ckpt* switch. Possible values for **occasion_specifier** are

n no checkpoint is performed.
s checkpoint when batch server is shut down.
m checkpoint at minimum CPU interval.
x checkpoint when job gets suspended.
<interval> checkpoint at specified time intervals.

The minimum CPU interval is defined in the queue configuration (see *sge_queue_conf(5)* for details). <interval> has to be specified in the format hh:mm:ss. The maximum of <interval> and the queue's minimum CPU interval is used if <interval> is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances. The <interval> will be available as a parameter with the name **c_interval**. The character sequence specified will be available as a parameter with the name **c_occasion**. Please note that if you change **c_occasion** via JSV, the last setting of **c_interval** will be overwritten and vice versa. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-ckpt ckpt_name

Available for *qsub* and *qalter* only.

Selects the checkpointing environment (see *sge_checkpoint*(5)) to be used for checkpointing the job. Also declares the job to be a checkpointing job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **ckpt**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-clear

Available for *qsub*, *qsh*, *qrsh*, and *qlogin* only.

Causes all elements of the job to be reset to their initial default status prior to applying any modifications (if any) appearing in this specific command.

-clearp parameter

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to clear list-based and non-list-based job parameters. As a result, the specified job parameter will be reset to the same default value that would have been used if the job were submitted with the *qsub* command without an additional specification

of **parameter** (e.g **-clearp N** would reset the job name to the default name. For script-based jobs this is the basename of the command script).

If a job is derived from a job class and if the access specifier that is defined before (or within a list-based attribute) does not allow deleting the parameter, the use of the **-clearp** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

The **parameter** argument might be either the name of a list-based job parameter as explained in the section **-adds** above, or a non-list parameter. Non-list parameter names are **a**, **A**, **ar**, **binding**, **ckpt**, **c_occasion**, **c_interval**, **dl**, **j**, **js**, **m**, **mbind**, **N**, **now**, **notify**, **P**, **p**, **pe_name**, **pe_range**, **r**, and **shell**.

-clears parameter key

Available for *qsub*, *qrsh* and *qalter* only.

Can also be used in combination with **-petask** to adjust PE task specific job requests.

Gives the user the ability to remove single entries of list-based job parameters including resource requests, job context, environment variables and more.

The **-adds** and **-mods** switches can be used to add or modify a single entry of a job parameter list.

If a job is derived from a job class and if the access specifier that is defined before or within a list-based attribute does not allow the removal of a specific entry from the list, the use of the **-clears** switch is disallowed for the corresponding entry. (See the **-jc** option below or find more information concerning job classes and access specifiers in sge_job_class*(5)*).

Parameter and **key** arguments are explained in more detail in the **-adds** section above.

-cwd

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the current working directory. This switch will activate Altair Grid Engine's path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

In the case of *qalter*, the previous definition of the current working directory will be overwritten if *qalter* is executed from a different directory from that of the preceding *qsub* or *qalter*.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

For *qrsh*, the **-cwd** and **-wd** switches are available only for qrsh calls in combination with a command. This means just calling qrsh -cwd is rejected, because in this case for interactive jobs, qrsh uses the login shell and changes into the specified login directory.

A command which allows the use of **-cwd** can be:

qrsh -cwd sleep 100 or qrsh -wd /tmp/ sleep 100

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **cwd**. The value of this parameter will be the absolute path to the working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result the job behaves as if **-cwd** were not specified during job submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-C prefix_string

Available for *qsub* and *qrsh* with script submission (**-b n**).

Prefix_string defines the prefix that declares a directive in the job's command. The prefix is not a job attribute, but affects the behavior of *qsub* and *qrsh*. If **prefix** is a null string, the command will not be scanned for embedded directives.
The directive prefix consists of two ASCII characters which, when appearing in the first two bytes of a script line, indicate that what follows is an Altair Grid Engine command. The default is "#$".
The user should be aware that changing the first delimiting character can produce unforeseen side effects. If the script file contains anything other than a "#" character in the first byte position of the line, the shell processor for the job will reject the line and may terminate the job prematurely.
If the -C option is present in the script file, it is ignored.

-dc variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Removes the given variable(s) from the job's context. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-display display_specifier

Available for *qsh* and *qrsh with command*.

Directs *xterm*(1) to use **display_specifier** in order to contact the X server. The **display_specifier** has to contain the hostname part of the display name (e.g. myhost:1). Local display names (e.g. :0) cannot be used in grid environments. Values set with the **-display** option overwrite settings from the submission environment and from **-v** command-line options.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **display**. This value will also be available in the job environment which may optionally be passed to JSV scripts. The variable name will be **DISPLAY**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-dl date_time

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the deadline initiation time in [[CC]YY]MMDDhhmm[.SS] format (see **-a** option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Altair Grid Engine administrator.

This option is applicable only for users allowed to submit deadline jobs.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **dl**. The format for the date_time value is CCYYMMDDhhmm.SS (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-e [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the path used for the standard error stream of the job. For *qsh*, *qrsh* and *qlogin* only the standard error stream of the prolog and epilog is redirected. If the **path** constitutes an absolute path name, the error-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard error stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default the file name for interactive jobs is */dev/null*. For batch jobs the default file name has the form *job_name._e_job_id* and *job_name._e_job_id*.**task_id** for array job tasks (See the **-t** option below).

If **path** is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded when the job runs and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently:

$HOME home directory on execution machine
$USER user ID of job owner
$JOB_ID current job ID
$JOB_NAME current job name (see **-N** option)
$HOSTNAME name of the execution host
$TASK_ID array job task index number

(The pseudo environment variable $TASK_ID is only available for array task jobs. If $TASK_ID is used and the job does not provide a task ID the resulting expanded string for $TASK_ID will be the text "undefined".)

Instead of $HOME, the tilde sign "~" can be used, as is common in *csh*(1) or *ksh*(1). Note that the "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires

corresponding permissions, of course. The "~" sign must be the first character in the path string.

If **path** or any component of it does not exist, it will be created with the permissions of the current user. A trailing "/" indicates that the last component of **path** is a directory. For example the command "qsub -e myjob/error.e $SGE_ROOT/examples/sleeper.sh" will create the directory "myjob" in the current working directory if it does not exist, and write the standard error stream of the job into the file "error.e". The command "qsub -e myotherjob /$SGE_ROOT/examples/sleeper.sh" will create the directory "myotherjob", and write the standard error stream of the job into a file with the default name (see description above). If it is not possible to create the directory (e.g. insufficient permissions), the job will be put in an error state.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **e**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hard

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all **-q** and **-l** resource requirements following in the command line, as well as in combination with **-petask** to specify PE task specific requests, will be hard requirements and must be satisfied in full before a job can be scheduled.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option (see below) is encountered during the scan, all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option or a corresponding value in *qmon* is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **l_hard**. Find more information in the sections describing **-q** and **-l**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-h | -h {u|s|o|n|U|O|S}...

Available for *qsub* (only **-h**), *qrsh*, *qalter* and *qresub* (hold state is removed when not set explicitly).

List of holds to place on a job, a task or some tasks of a job.

'u' denotes a user hold.
's' denotes a system hold.
'o' denotes an operator hold.
'n' denotes no hold (requires manager privileges).

As long as any hold other than 'n' is assigned to the job, the job is not eligible for execution. Holds can be released via *qalter* and *qrls*(1). *qalter* can be used to do this via the following additional option specifiers for the **-h** switch:

'U' removes a user hold.
'S' removes a system hold.
'O' removes an operator hold.

Altair Grid Engine managers can assign and remove all hold types, Altair Grid Engine operators can assign and remove user and operator holds, and users can only assign or remove user holds.

When using *qsub*, only user holds can be placed on a job and thus only the first form of the option with the **-h** switch is allowed. As opposed to this, *qalter* requires the second form described above.

An alternate means to assign hold is provided by the *qhold*(1) facility.

If the job is a array job (see the **-t** option below), all tasks specified via **-t** are affected by the **-h** operation simultaneously.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified with *qsub* or during the submission of a job in *qmon*, the parameter **h** with the value **u** will be passed to the defined JSV instances indicating that the job will have a user hold after the submission finishes. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).) Within a JSV script it is possible to set all above described hold states (also in combination) depending on user privileges.

-help

Prints a listing of all options.

-version

Display version information for the command.

-hold_jid wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. The submitted job is not eligible for execution unless all jobs referenced in the comma-separated job ID and/or job name list have completed. If any of the referenced jobs exit with exit code 100, the submitted job will remain ineligible for execution.

With the help of job names or a regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name

dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-hold_jid_ad wc_job_list

Available for *qsub*, *qrsh*, and *qalter* only. See *sge_types*(1) for **wc_job_list** definition.

Defines or redefines the job array dependency list of the submitted job. A reference by job name or pattern is only accepted if the referenced job is owned by the same user as the referring job. Each sub-task of the submitted job is not eligible for execution unless the corresponding sub-tasks of all jobs referenced in the comma-separated job ID and/or job name list have completed. If any array task of the referenced jobs exit with exit code 100, the dependent tasks of the submitted job will remain ineligible for execution.

With the help of job names or regular pattern one can specify a job dependency on multiple jobs satisfying the regular pattern or on all jobs with the requested names. The name dependencies are resolved at submit time and can only be changed via qalter. New jobs or name changes of other jobs will not be taken into account.

If either the submitted job or any job in wc_job_list are not array jobs with the same range of sub-tasks (see **-t** option below), the request list will be rejected and the job creation or modification will fail.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **hold_jid_ad**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-i [[hostname]:]file,...

Available for *qsub* and *qalter* only.

Defines or redefines the file used for the standard input stream of the job. If the *file* constitutes an absolute filename, the input-path attribute of the job is set to **path**, including the **hostname**. If the path name is relative, Altair Grid Engine expands **path** either with the current working directory path (if the **-cwd** switch (see above) is also specified) or with the home directory path. If **hostname** is present, the standard input stream will be placed in the corresponding location only if the job runs on the specified host. If the path contains a ":" without a **hostname**, a leading ":" has to be specified.

By default /dev/null is the input stream for the job.

It is possible to use certain pseudo variables, whose values will be expanded at the runtime of the job and will be used to express the standard input stream as described in the *-e* option for the standard error stream.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **i**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-inherit

Available only for *qrsh* and *qmake*(1).

qrsh allows the user to start a task in an already-scheduled parallel job. The option **-inherit** tells *qrsh* to read a job ID from the environment variable JOB_ID and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command will be executed must precede the command to execute; the syntax changes to

qrsh -inherit [**other options**] **hostname command** [**command_args**]

Note also, that in combination with **-inherit**, most other command-line options will be ignored. Only the options **-verbose**, **-v** and **-V** will be interpreted. As a replacement to option **-cwd**, please use **-v PWD**.

Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option **-V** should be suitable for most applications.

With **-inherit**, *qrsh* also tries to read the environment variable SGE_PE_TASK_CONTAINER_REQUEST, which allows specifying the ID of the parallel task container in which this newly-started task shall run. This is useful if per parallel task specific requests (see **-petask** option) were used to assign specific resources to this parallel task container; these are resources which this task needs to run. Please be aware the specified parallel task container must exist on the specified host and must still be unused, otherwise starting this task will fail.

In the task itself, the environment variable SGE_PE_TASK_CONTAINER_ID is set to the ID of the PE task container in which the task was started. Furthermore, the environment variable SGE_PE_TASK_ID is set to the unique ID of this parallel job task. The SGE_PE_TASK_ID has the format ".", where n is a consecutive number of tasks of the current job that has been started on the specified host.

Note: If in your system the qmaster TCP port is not configured as a service, but rather via the environment variable SGE_QMASTER_PORT, make sure that this variable is set in the environment when calling *qrsh* or *qmake* with the **-inherit** option. If you call *qrsh* or *qmake* with the **-inherit** option from within a job script, export SGE_QMASTER_PORT with the option "-v SGE_QMASTER_PORT" either as a command argument or as an embedded directive.

This parameter is not available in the JSV context. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-j y[es]|n[o]

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies whether or not the standard error stream of the job is merged into the standard output stream.
If both the **-j y** and the **-e** options are present, Altair Grid Engine sets but ignores the error-path attribute.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **j**. The value will also be **y** when the long form **yes** was specified during submission. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jc jc_name

Available for *qsub*, *qrsh*, and *qalter* only.

Indicates whether the job specification of a job should be derived from a job class. **jc_name** might be either a name of a job class or the combination of a job class name and a variant name, with both names separated by a dot (.).

If this switch is used, the following 6 steps will be executed within the *sge_qmaster*(8) process:

(1) A new job will be created

(2) This job structure will be initialized with default values.

(3) Then all those default values will be replaced with the values that are specified in job template attributes in the job class (or job class variant).

(4) If the **-jc** switch was combined with other command-line switches that specify job characteristics, those settings will be applied to the job. This step might overwrite default values and values that where copied from the job class specification.

(5) Server JSV will be triggered if configured. This server JSV script will receive the specification of the job and if the server JSV adjusts the job specification, default values, values derived from the job class specification and values specified at the command line might be overwritten.

(6) With the last step sge_qmaster checks whether any access specifiers were violated during steps (4) or (5). If this is the case, the job is rejected. Otherwise it enters the list of pending jobs.

The server JSV that might be triggered with step (5) will receive the **jc_name** as a parameter with the name **jc**. If a server JSV decides to change the **jc** attribute, the process described above will restart at step (1) and the new **jc_name** will be used for step (3).

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

Any attempt to change a job attribute of a job that was derived from a job class will be rejected. Owners of the job class can soften this restriction by using access specifiers within the specification of a job class. Details concerning access specifiers can be found in *sge_job_class*(5.)

The qalter -jc NONE command can be used by managers to release the link between a submitted job class job and its parent job class. In this case all other job parameters won't be changed but it will be possible to change all settings with qalter afterwards independent of the access specifiers that were used.

-js job_share

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the job share of the job relative to other jobs. Job share is an unsigned integer value. The default job share value for jobs is 0.

The job share influences the Share Tree Policy and the Functional Policy. It has no effect on the Urgency and Override Policies (see *sge_share_tree*(5), *sge_sched_conf*(5) and the *Altair Grid Engine Installation and Administration Guide* for further information on the resource management policies supported by Altair Grid Engine).

When using the Share Tree Policy, users can distribute the tickets to which they are currently entitled among their jobs using different shares assigned via **-js**. If all jobs have the same job share value, the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. In this case, job shares are treated like an additional level in the share tree.

In connection with the Functional Policy, the job share can be used to weight jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category.

If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies, and the tickets independently derived in each of them are added to the total number of tickets for each job.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **js**. (See the **-jsv** option below or find more information concerning JSV in *sge_jsv*(5).)

-jsv jsv_url

Available for *qsub*, *qsh*, *qrsh* and *qlogin* only.

Defines a client JSV instance which will be executed to verify the job specification before the job is sent to qmaster.

In contrast to other options this switch will not be overwritten if it is also used in sge_request files. Instead all specified JSV instances will be executed to verify the job to be submitted.

The JSV instance which is directly passed with the command line of a client is executed first to verify the job specification. After that the JSV instance which might have been defined in

various sge_request files will be triggered to check the job. Find more details in man page *sge_jsv*(5) and *sge_request*(5).

The syntax of the **jsv_url** is specified in *sge_types(1).*()

-masterl resource=value,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. Available only in combination with parallel jobs.

Launch the parallel job in a Altair Grid Engine queue that meets the given resource request list for the master task of that parallel job. Other resource requests that can be specified with the l-switch will only specify the requirements of agent tasks as long as the masterl-switch is used during job submission.

Using **-masterl** is not advised any longer; instead use **-petask controller -l ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

When using *qalter* the previous definition is replaced by the specified one.

sge_complex(5) describes how a list of available resources and their associated valid value specifiers can be obtained.

qalter does allow changing the value of this option while the job is running; however the modification will only be effective after a restart or migration of the job.

If this option or a corresponding value in *qmon* is specified, the hard resource requirements will be passed to defined JSV instances as a parameter with the name **masterl**. If regular expressions will be used for resource requests,, these expressions will be passed as they are. Also shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-l resource=value,... (or -l resource=value -l ...)

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Launches the job in a Altair Grid Engine queue instance meeting the given resource request list.

There may be multiple **-l** switches specified to define the desired queue instance. When using qalter the previous definition is completely replaced by the specified one but also here **-l** might be specified multiple times.

```
1) -l arch=lx-amd64 -l memory=4M
```

Requests a Linux queue instance that provides 4M of memory

```
2) -l arch=lx-amd64,memory=4M
```

Same as 1)

Can be combined with **-soft**/**-hard** to define soft and hard resource requirements. If the resource request is specified while the **-soft** option is active the value for consumables can also be specified as a range. (See the *sge_complex*(5) for more details).

```
3) -l arch=lx-amd64 -soft -l memory=4M-8M:1M -l lic=1
```

Requests a Linux queue instance (as an implicit hard request) with a soft memory request and an optional software license.

Can be combined with the **-petask** switch to define resource requests for individual tasks or a group of tasks where the common resource requests should not apply or should be different.

```
4) -pe pe1 8 -l memory=4M
```

PE job with 8 tasks where each task requests 4M memory

```
5) -pe pe1 8 -petask contoller -l memory=4M
              -petask 1 -l memory=1M
```

PE job with 8 tasks. Only 2 tasks have memory requests. Memory requests for the controller task (which has ID 0) and task 1 are different.

```
6) -pe pe1 4-8 -l memory=1M -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory. All other remaining tasks require only 1M.

```
7) -pe pe1 4-8 -l memory=1M -q A.q -petask 1-:2 -l memory=2M
```

Every second PE task requests 2M of memory and no queue. All common requests are only valid for those tasks that have no explicit requests. As a result all tasks with even task numbers are bound to the A.q whereas ueven tasks can be executed in any queue.

qalter allows changing the value of this option even while the job is running. If **-when now** is set the changes will be done immediately if possible (only for consumables). If **-when on_reschedule** (default) is set the changes take effect when the job gets restarted or is migrated.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft** for a specific job variant. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests„ these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1)) If this option or a corresponding value in *qmon* is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **l_hard** and **l_soft**. If regular expressions are used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-m b|e|a|s|n,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the **-M** option described below. The option arguments have the following meaning:

'b' Mail is sent at the beginning of the job.
'e' Mail is sent at the end of the job.
'a' Mail is sent when the job is aborted or rescheduled.
's' Mail is sent when the job is suspended.
'n' No mail is sent.

Currently no mail is sent when a job is suspended.

qalter allows changing the b, e, and a option arguments even while the job executes. The modification of the b option argument will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **m**. (See the **-jsv** option above or find more information concerning JSV in

-M user[@host],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host.

qalter allows changing this option even while the job executes.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **M**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-masterq wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*. Only meaningful for parallel jobs, i.e. together with the -pe option.

Defines or redefines a list of cluster queues, queue domains and queue instances which may be used to become the so-called *master queue* of this parallel job. A more detailed description of *wc_queue_list* can be found in *sge_types*(1). The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*.

Using **-masterq** is not advised any longer; instead use **-petask controller -q ...**. It is not permissible to combine the **-petask** switch with either **-masterq** or **-masterl**.

Depending on the requested *master queue* and other queue requests, whether implicit or explicit, depending on how the queues are spread out over the execution hosts and depending on the allocation_rule of the parallel environment, requesting a *master queue* may make it necessary for Altair Grid Engine to allocate one task more for this parallel job than the user requested. See *sge_pe*(5), "allocation_rule" for details.

This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this hard resource requirement will be passed to defined JSV instances as a parameter with the name **masterq**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-mods parameter key value

Available for *qsub*, *qrsh*, *qalter* of Altair Grid Engine only.

Can also be used in combination with **-petask** to adjust PE task specific requests of a job.

Gives the user the ability to modify entries of list-based job parameters such as resource requests, job context, environment variables and more.

The **-adds** and **-clears** switches can be used to add or remove a single entry of a job parameter list.

Parameter, **key** and **value** arguments are explained in more detail in the **-adds** section above.

-mbind

Available for *qsub*, *qrsh*, and *qalter*. Supported on lx-amd64 execution hosts only (for more details try **utilbin/loadcheck -cb** on the execution host).

Sets the memory allocation strategy for all processes and sub-processes of a job. On execution hosts with a NUMA architecture, the memory access latency and memory throughput depends on which NUMA node the memory is allocated and on which socket/core the job runs. In order to influence the memory allocation different submit options are provided:

-mbind cores Prefers memory on the NUMA node where the job is bound with core binding. Requires core binding set with -binding. The optional "m_mem_free" request is enhanced during scheduling time with implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. For more details see **-mbind cores:strict**

-mbind cores:strict The job is only allowed to allocate memory on the NUMA node to which it is bound. Requires core binding set with -binding. The optional "m_mem_free" request is extended during scheduling time by implicit per NUMA node requests ("m_mem_free_n<node>"), depending on which cores the job is bound to. The amount of selected cores per NUMA node and the total memory per slot request determine the amount of required memory per NUMA node. Hence when using the "m_mem_free"

memory request the job is only scheduled onto sockets which offer the specified amount of free memory.

-mbind round_robin Sets the memory allocation strategy for the job to interleaved memory access. When the memory resource request "m_mem_free" is used, the scheduler also adds implicit memory requests for all NUMA nodes on the execution host ("m_mem_free_n<node>").

-mbind nlocal Only allowed for serial jobs or jobs using a parallel environment with allocation rule "$pe_slots". Unspecified behavior for other PEs. Automatically binds a sequential or multi-threaded job to cores or sockets and sets an appropriate memory allocation strategy. Requires a resource request for the "m_mem_free" host complex. The behavior of the scheduler depends on the execution hosts characteristics as well as on whether the job is a serial or a multi-threaded parallel job (PE job with allocation rule "$pe_slots"). It is not permissible to override the implicit core binding with the **-binding** switch.

The scheduler algorithm for serial jobs is as follows:
- If the host can't fulfill the "m_mem_free" request, the host is skipped.
- If the job requests more RAM than is free on each socket but less than is installed on the sockets, the host is skipped.
- If the memory request is smaller than the amount of free memory on a socket, it tries to bind the job to "one core on the socket" and decrements the amount of memory on this socket ("m_mem_free_n<nodenumber>"). The global host memory "m_mem_free" on this host is decremented as well.
- If the memory request is greater than the amount of free memory on any socket, it finds an unbound socket and binds it there completely, and allows memory overflow. Decrements from "m_mem_free" as well as from "m_mem_free_n<socketnumber>", then decrements the remaining memory in round-robin fashion from the remaining sockets. . If the scheduler does not find enough free memory, it goes to the next host.

The scheduler algorithm for parallel jobs is as follows:
- Hosts that don't offer "m_mem_free" memory are skipped (of course hosts that don't offer the amount of free slots requested are skipped as well).
- If the amount of requested slots is greater than the amount of cores per socket, the job is dispatched to the host without any binding.
- If the amount of requested slots is smaller than the amount of cores per socket, it does following:
- If there is any socket which offers enough memory ("m_mem_free_n<N>") and enough free cores, binds the job to these cores and sets memory allocation mode to "cores:strict" (so that only local memory requests can be done by the job).
- If this is not possible it tries to find a socket which is completely unbound and has more than the required amount of memory installed ("m_mem_total_n<N>"). Binds the job to the complete socket, decrements the memory on that socket at "m_mem_free_n<N>" (as well as host globally on "m_mem_free") and sets the memory allocation strategy to "cores" (preferred usage of socket local memory).

If the parameters request the "m_mem_free" complex, the resulting NUMA node memory requests can be seen in the "implicit_requests" row in the qstat output.

Note that resource reservations for implicit per NUMA node requests as well as topology selections for core binding are not part of resource reservations yet.

The value specified with the **-mbind** option will be passed to defined JSV instances (as

"mbind") only when set. JSV can set the parameter as "round_robin", "cores", "cores:strict", or "NONE". The same values can be used for job classes.

-notify

Available for *qsub*, *qrsh* (with command) and *qalter* only.

This flag when set causes Altair Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending, the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending, the job will receive a SIGUSR2 several seconds before the SIGKILL. This option provides the running job, before receiving the SIGSTOP or SIGKILL, a configured time interval to do e.g. cleanup operations. The amount of time delay is controlled by the **notify** parameter in each queue configuration (see *sge_queue_conf(5)*).

Note that the Linux operating system "misused" the user signals SIGUSR1 and SIGUSR2 in some early Posix thread implementations. You might not want to use the **-notify** option if you are running multi-threaded applications in your jobs under Linux, particularly on 2.0 or earlier kernels.

qalter allows changing this option even while the job executes.

Only if this option is used the parameter named **notify** with the value **y** will it be passed to defined JSV instances. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-now y[es]|n[o]

Available for *qsub*, *qsh*, *qlogin* and *qrsh*.

-now y tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure or if the job could not be scheduled immediately. For array jobs submitted with the **-now** option, if one or more tasks can be scheduled immediately the job will be accepted; otherwise it will not be started at all.

Jobs submitted with **-now y** option can ONLY run on INTERACTIVE queues. **-now y** is the default for *qsh*, *qlogin* and *qrsh*

With the **-now n** option, the job will be put into the pending queue if it cannot be executed immediately. **-now n** is default for *qsub*.

The value specified with this option, or the corresponding value specified in *qmon*, will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **now**. The value for this parameter will be **y** when the long form **yes** was specified during submission. Please note that the parameter within JSV is a read-only parameter that cannot be changed. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-N name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The name of the job. The name should follow the "**name**" definition in *sge_types*(1). Invalid job names will be denied at submit time.

If the **-N** option is not present, Altair Grid Engine assigns the name of the job script to the job after any directory pathname has been removed from the script-name. If the script is read from standard input, the job name defaults to STDIN.

When using *qsh* or *qlogin* without the **-N** option, the string 'INTERACT' is assigned to the job.

In the case of *qrsh* if the **-N** option is absent, the resulting job name is determined from the qrsh command line by using the argument string up to the first occurrence of a semicolon or whitespace and removing the directory pathname.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will be passed to defined JSV instances as a parameter with the name *N*. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-noshell

Available only for *qrsh* when used with a command line.

Do not start the command line given to *qrsh* in a user's login shell, i.e., execute it without the wrapping shell.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the **-noshell** option or include the shell call in the command line.

Example:

```
qrsh echo '$HOSTNAME'
Alternative call with the -noshell option
qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'
```

-nostdin

Available only for *qrsh*.

Suppresses the input stream STDIN. *qrsh* will pass the option -n to the *rsh*(1) command. This is especially useful when multiple tasks are executed in parallel using *qrsh*, e.g. in a *qmake*(1) process, where it would be undefined as to which process would get the input.

-o [[hostname]:]path,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

The path used for the standard output stream of the job. The **path** is handled as described in the **-e** option for the standard error stream.

By default the file name for standard output has the form *job_name._o_job_id* and *job_name._o_job_id*.**task_id** for array job tasks (see **-t** option below).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **o**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-ot override_tickets

Available for *qalter* only.

Changes the number of override tickets for the specified job. Requires manager/operator privileges.

-P project_name

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See the **-aprj** option to *qconf*(1)).

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **P**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-p priority

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024, -1023 is the lowest priority, 1024 the highest priority. The default priority value for jobs is 0.

Users may only use the priority range from -1023 to 0 in job submission, managers and operators may use the full range from -1023 to 1024 in job submission.

By default, users may only decrease the priority of their jobs. If the parameter **ALLOW_INCREASE_POSIX_PRIORITY** is set as **qmaster_param** in the global configuration, users are also allowed to increase the priority of their own jobs up to 0.

Altair Grid Engine managers and operators may also increase the priority associated with jobs independent from *ALLOW_INCREASE_POSIX_PRIORITY* setting.

If a pending job has higher priority, that gives it earlier eligibility to be dispatched by the Altair Grid Engine scheduler.

If this option or a corresponding value in *qmon* is specified and the priority is not 0, this value will be passed to defined JSV instances as a parameter with the name **p**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-par allocation_rule

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only. This option can be used with parallel jobs only.

It can be used to overwrite the allocation rule of the parallel environment a job gets submitted into with the **-pe** submit option. The specified allocation rule will be used for scheduling the parallel job.

Can also be used after a **-petask** switch, but only to overwrite the allocation rule of the controller task and only to set it to the allocation rule "1". E.g.: $ qsub -pe pe_slots.pe 4 -petask controller -par 1 job.sh This will put the controller task on one host and three agent tasks on a different host.

Valid allocation rules are **$pe_slots** , **$fill_up** , **$round_robin** and positive numbers as **fixed allocation rule**

See also the section "**allocation_rule**" in *sge_pe*(5).

If this option is specified its value will be passed to defined JSV instances as a parameter with the name **par**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-pe parallel_environment n[-[m]]|[-]m,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Parallel programming environment (PE) to instantiate. For more detail about PEs, please see the *sge_types*(1).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, the parameters **pe_name**, **pe_min** and **pe_max** will be passed to configured JSV instances where **pe_name** will be the name of the parallel environment and the values **pe_min** and **pe_max** represent the values n and m which have been provided with the **-pe** option. A missing specification of m will be expanded as value 9999999 in JSV scripts and it represents the value infinity.

Since it is possible to specify more than one range with the **-pe** option, the JSV instance **pe_n** will contain the number of specified ranges. The content of of the ranges can be addressed by adding the index to the variable name. The JSV variable **pe_min_0** represents the first minimum value of the first range.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-petask tid_range_list ...

Available for *qsub*, *qrsh* (with command) and *qalter*.

Can be used to submit parallel jobs (submitted with **-pe** request); this signifies that all resource requirements specified with **-q** and **-l**, and in combination with **-hard** and **-soft**, that follow in the command line, will be requirements for the parallel tasks selected by **tid_range_list**. Can also be used in combination with **-adds**, **-mods**, **-clears** and **-clearp** to adjust parameters of a job or a job that was derived from a JC either during submission with one of the submit clients or later on with *qalter*.

For requests for the PE task 0 and only PE task 0, not for ranges that contain the PE task 0, it is also possible to use the **-par** switch with the allocation_rule "1" in order to force the controller task to a different host from those of all agent tasks.

As soon as a task is specified within one **tid_range_list** with a **-pe_task** switch, all other resource requests outside the scope of any **-pe_task** switch will be invalidated for that specific task.

tid_range_list is a single range expression, a comma-separated list of range expressions of the form n[-[m][:s]], or a keyword that represents one or multiple PE tasks. Find more details concerning **tid_range_list** in *sge_types*(1).

```
1) -petask 0 ...
2) -petask controller ...
```

Both denote the controller task.

```
3) -petask 1- ...
4) -petask 1-:1 ...
5) -petask agents ...
```

All three expressions are equivalent and specify all tasks beginning from task 1, meaning all except for task 0.

```
6) -petask 1-:2 ...
```

All tasks with uneven task numbers.

```
7) -petask 1-9:2,10,12 ...
```

The tasks 1, 3, 5, 7, 9, 10 and 12.

The **-petask** switch can be used within one command line multiple times to specify different resource requests for individual tasks or sets of tasks. Each command line that contains ambiguous resource specifications for tasks will be rejected. This means that task ID and range specifications within one **-petask** construct as well as all specifications of multiple -petask specifications for a job variant need to be non-overlapping.

```
8) -petask 1-9:2 ... -petask 6,8,9 ...
```

Would be rejected because task with ID 9 is part of two different **-petask** specifications.

For tasks where no explicit resource requests are specified as part of a **-petask** request the common resource requests specified outside the scope of a -petask switch apply.

```
9) -pe pe 8 -l memory=1M
            -petask controller -l memory=4M
            -petask 2-8:2 -l memory=2G
```

The controller task (ID 0) has a memory request of 4M. All agent tasks with even IDs request 2G of memory. All other remaining tasks not explicitly specified (here 1, 3, 5, 7) have a 1M memory request.

```
10) -pe pe 8 -l memory=1M -q A.q
             -petask 1 -l memory=2G
```

For all tasks the queue request of A.q will be valid except for task 1. This task has an explicit request specified and the absence of an explicit queue request will overwrite the common request that is outside of the scope of any -petask switch.

It is not allowed to combine the **-petask** switch with either **-masterq** or **-masterl**.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are only there to start the parallel program, are not part of the parallel program itself, and are not considered to consume a significant amount of CPU time, memory and other resources. Therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each individual task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less than the number of requested PE tasks for this job.

```
11)
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

These submit command lines request a special queue for their last PE task:

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

qalter can be used in combination with the **-petask** switch to completely replace the previously-specified requests for an existing task ID range as long as no additional restrictions apply. Find more details for restrictions in the corresponding sections for **-l** and **-q** of this man page.

It is also possible to adjust parts of parallel task specific requests in combination with the **-add**, **-mods**, and **-clears** switches, especially if a job was initially created using a job class specified with the **-jc** switch.

Attempts to create new requests for a task or set of tasks that were not also specified during submission of a job, or attempts to change requests for a subset of tasks that are part of a task range that was specified during job submission, will be rejected later on by **qalter**.

Task ranges have to be formed during submission or within JSV instances before a job is initially accepted.

Common hard and soft resource requests as well as attempts to overwrite the parallel allocation rule of parallel jobs (all requests *not* following a **-petask** switch) will be passed to defined JSV instances as parameters named *l_hard*, *l_soft*, *q_hard*, *q_soft*, and *par*. They are valid for those tasks of a parallel job that have no task-specific requests.

Task-specific resource request information including the task range information for which those requests are valid is passed to JSV as follows. *petask_max* provides the information on how many task specific requests were specified. *petask_<id>_ranges* shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters *petask_<id>_<range>_min*, *petask_<id>_<range>_max* and *petask_<id>_<range>_step* where the corresponding <id> and <range> denote the corresponding position. Numbering starts with 0 and <id> and <range> may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluating the parameters *petask_<id>_l_hard*, *petask_<id>_l_soft*, *petask_<id>_q_hard*, *petask_<id>_q_soft* and *petask_<id>_l_par*; the latter only for the controller task and only with allocation_rule “1”.

In case range specifications should be reduced within JSVs (IDs should be removed from ranges or range elements from lists) the writer of the JSV script has to ensure that the parameter values that define the maximum amount of task requests (*petasks_max*), ranges for task requests (*petask_<id>_ranges*) and min, max and step-size values are set correctly before *jsv_correct()* is called.

Range specifications (*petask_<id>_<range>_...*) and task-specific requests (*petask_<id>_...*) for tasks that exceed the defined maximum values (cannot be deleted within the JSV) will be automatically be ignored when *jsv_correct()* is called to transfer job modifications from the JSV to the submit-client or qmaster (client or server JSV).

The same restrictions apply to JSV implementations as those that apply to range-specific requests at the command line. This means task ranges are not allowed to overlap each other and only one range with an open end is allowed in a range specification for one variant of a job; otherwise the job will be rejected.

If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the -jsv option above or find more information concerning JSV in jsv(1))

-pty y[es]|n[o]

Available for *qrsh*, *qlogin* and *qsub* only.

-pty yes forces the job to be started in a pseudo terminal (pty). If no pty is available, the job start fails. -pty no forces the job to be started without a pty. By default, *qrsh without a command* or *qlogin* start the job in a pty, and *qrsh with a command* or *qsub* start the job without a pty.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **pty** will be available and it will have the value **u** when the switch was omitted or the value **y** or **n** depending on whether **y[es]** or **n[o]** was passed as a parameter with the switch. This parameter can be changed in the JSV context to influence the behavior of the command-line client and job.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-q wc_queue_list

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Defines or redefines a list of cluster queues, queue domains, or queue instances which may be used to execute a job. Please find a description of *wc_queue_list* in *sge_types*(1). This parameter has all the properties of a resource request and will be merged with requirements derived from the **-l** option described above.

Can be combined with **-hard** and **-soft** to define soft and hard queue requirements, and can also be combined with **-petask** to specify specific queue requirements for individual PE tasks or group of PE tasks. Find more information in the corresponding sections of this man page.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option is specified, these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names **q_hard** and **q_soft**. PE task specific requests as well as information about the tasks where those requests are applied have the additional prefix **petask_<id>_*** in corresponding names. If regular expressions will be used for resource requests, these expressions will be passed as they are. In addition, shortcut names will not be expanded. (See the **-jsv** option above or find more information concerning JSV in sge_jsv_(1))

-R y[es]|n[o]

Available for *qsub*, *qrsh*, *qsh*, *qlogin* and *qalter*.

Indicates whether reservation for this job should be done. Reservation is never done for immediate jobs, i.e. jobs submitted using the **-now yes** option. Please note that regardless of the reservation request, job reservation might be disabled using max_reservation in *sge_sched_conf*(5) and might be limited only to a certain number of high-priority jobs.

By default jobs are submitted with the **-R n** option.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **R**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-r y[es]|n[o]

Available for *qsub* and *qalter* only.

Identifies the ability of a job to be rerun or not. If the value of **-r** is 'yes', the job will be rerun if the job was aborted without leaving a consistent exit state. (This is typically the case when the node on which the job is running crashes). If **-r** is 'no', the job will not be rerun under any circumstances.
Interactive jobs submitted with *qsh, qrsh* or *qlogin* are not rerunnable.

qalter allows changing this option even while the job executes.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **r**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-rou variable,...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Used to specify which job report attributes (e.g. cpu, mem, vmem, ...) shall get written to the reporting file and the reporting database.

Variables are specified as a comma-separated list.

Specifying reporting variables per job will overwrite a global setting done in the global cluster configuration named reporting_params; see also *sge_conf*(5).

-rdi y[es]|n[o]

Available for *qsub* and *qalter* only.

This parameter is shorthand for **request dispatch information** and is used to specify jobs for which messages should be collected when the *sge_sched_conf*(5) **schedd_job_info** parameter is set to **if_requested**. Setting the **-rdi yes** option will allow the *qstat -j* command to print reasons why the job cannot be scheduled. The option can also be set or changed after a job has been submitted using the *qalter* command. The default option is **-rdi no**.

-sc variable[=value],...

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Sets the given name/value pairs as the job's context. **Value** may be omitted. Altair Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple **-ac**, **-dc**, and **-sc** options may be given. The order is important. The variable name must not start with the characters "+", "-" or "=".

Contexts provide a way to dynamically attach and remove meta-information to and from a job. The context variables are **not** passed to the job's execution context in its environment.

qalter allows changing this option even while the job executes.

The outcome of the evaluation of all **-ac**, **-dc**, and **-sc** options or corresponding values in *qmon* is passed to defined JSV instances as a parameter with the name **ac**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-shell y[es]|n[o]

Available only for *qsub*.

-shell n causes qsub to execute the command line directly, as if by *exec*(2). No command shell will be executed for the job. This option only applies when **-b y** is also used. Without **-b y**, **-shell n** has no effect.

This option can be used to speed up execution as some overhead, such as starting the shell and sourcing the shell resource files, is avoided.

This option can only be used if no shell-specific command line parsing is required. If the command line contains shell syntax such as environment variable substitution or (back) quoting, a shell must be started. In this case either do not use the **-shell n** option or execute the shell as the command line and pass the path to the executable as a parameter.

If a job executed with the **-shell n** option fails due to a user error, such as an invalid path to the executable, the job will enter the error state.

-shell y cancels the effect of a previous **-shell n**. Otherwise, it has no effect.

See **-b** and **-noshell** for more information.

The value specified with this option or the corresponding value specified in *qmon* will only be passed to defined JSV instances if the value is *yes*. The name of the parameter will be **shell**. The value will be **y** when the long form **yes** was specified during submission. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-si session_id

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Requests sent by this client to the *sge_qmaster*(8) daemon will be done as part of the specified session. If the switch is omitted or if **NONE** is specified as **session_id**, such requests will be executed outside the control of a session.

Find more information concerning sessions in *sge_session_conf*(5).

-soft

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

Signifies that all resource requirements (**-l** and **-q**) following in the command line, and in combination with **-petask** to specify PE task specific requests, will be soft requirements and are to be filled on an “as available” basis.

It is possible to specify ranges for consumable resource requirements if they are declared as **-soft** requests. More information about soft ranges can be found in the description of the **-l** option.

As Altair Grid Engine scans the command line and script file for Altair Grid Engine options and parameters, it builds a list of resources required by the job. All such resource requests are considered as absolutely essential for the job to commence. If the **-soft** option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice to have, but not essential". If the **-hard** flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The **-hard** and **-soft** options in effect act as "toggles" during the scan.

If this option is specified, the corresponding **-q** and **-l** resource requirements will be passed to defined JSV instances as parameters with the names **q_soft** and **l_soft**. PE task specific requests as well as information about the tasks where those requests are to be applied have the additional prefix **petask_<id>_*** in corresponding names. Find more information in the sections describing **-q** and **-l**. (see **-jsv** option above or find more information concerning JSV in sge_jsv_(1)).

-sync y|n|l|r

Available for *qsub*.

-sync y causes *qsub* to wait for the job to complete before exiting. If the job completes successfully, *qsub's* exit code will be that of the completed job. If the job fails to complete successfully, *qsub* will print out an error message indicating why the job failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, the job will be canceled.
With the **-sync n** option, *qsub* will exit with an exit code of 0 as soon as the job is submitted successfully. **-sync n** is default for *qsub*.
If **-sync y** is used in conjunction with **-now y**, *qsub* will behave as though only **-now y** were given until the job has been successfully scheduled, after which time *qsub* will behave as though only **-sync y** were given.
If **-sync y** is used in conjunction with **-t n[-m[:i]]**, *qsub* will wait for all the job's tasks to complete before exiting. If all the job's tasks complete successfully, *qsub's* exit code will be that of the first completed job task with a non-zero exit code, or 0 if all job tasks exited with an exit code of 0. If any of the job's tasks fail to complete successfully, *qsub* will print out an error message indicating why the job task(s) failed and will have an exit code of 1. If *qsub* is interrupted, e.g. with CTRL-C, before the job completes, all of the job's tasks will be canceled.
With the **-sync l** option, *qsub* will print an appropriate message as soon as the job changes into the l-state (license request sent to License Orchestrator).
With the **-sync r** option, *qsub* will print an appropriate message as soon as the job is running.

All those options can be combined. qsub will exit when the last event occurs.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **sync** will be available and it will have the value **y** when the switch was used. The parameter value cannot be changed within the JSV context.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-S [[hostname]:]pathname,...

Available for *qsub*, *qsh*, and *qalter*.

Specifies the interpreting shell for the job. Only one **pathname** component without a **host** specifier is valid and only one pathname for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list.

Furthermore, the pathname can be constructed with pseudo environment variables as described for the **-e** option above.

When using *qsh* the specified shell path is used to execute the corresponding command interpreter in the *xterm*(1) (via its *-e* option) started on behalf of the interactive job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as a parameter with the name **S**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-t n[-m[:s]]

Available for *qsub* only. *qalter* cannot be used to change the array job size but **-t** might be used in combination with a job ID to address the tasks that should be changed.

Submits a so called *Array Job*, i.e. an array of identical tasks differentiated only by an index number and treated by Altair Grid Engine almost like a series of jobs. The option argument to **-t** specifies the number of array job tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable **SGE_TASK_ID**. The option arguments n, m and s will be available through the environment variables **SGE_TASK_FIRST**, **SGE_TASK_LAST** and **SGE_TASK_STEPSIZE**.

The following restrictions apply to the values n and m:

```
1 <= n <= MIN(2^31-1, max_aj_tasks)
1 <= m <= MIN(2^31-1, max_aj_tasks)
n <= m
```

max_aj_tasks is defined in the cluster configuration (see *sge_conf*(5)).

The task ID range specified in the option argument may be a single number, a simple range of the form n-m, or a range with a step size. Hence the task ID range specified by 2-10:2 will result in the task ID indexes 2, 4, 6, 8, and 10, for a total of 5 identical tasks, each with the environment variable SGE_TASK_ID containing one of the 5 index numbers.

All array job tasks inherit the same resource requests and attribute definitions specified in the *qsub* or *qalter* command line, except for the **-t** option. The tasks are scheduled independently and, provided enough resources exist, concurrently, very much like separate jobs. However, an array job or a sub-array thereof can be accessed as a single unit by commands like *qmod*(1) or *qdel*(1). See the corresponding manual pages for further detail.

Array jobs are commonly used to execute the same type of operation on varying input data sets where each data set is associated with a task index number. The number of tasks in an array job is unlimited.

STDOUT and STDERR of array job tasks will be written into different files with the default location

<jobname>.['e'|'o']<job_id>'.'<task_id>

In order to change this default, the **-e** and **-o** options (see above) can be used together with the pseudo environment variables $HOME, $USER, $JOB_ID, $JOB_NAME, $HOSTNAME, and $TASK_ID.

Note that while you can use the output redirection to divert the output of all tasks into the same file, the result of this is undefined.

If this option or a corresponding value in *qmon* is specified, this value will be passed to defined JSV instances as parameters with the names **t_min**, **t_max**, and **t_step**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tc max_running_tasks

Available for *qsub* and *qalter* only.

Can be used in conjunction with array jobs (see -t option) to set a self-imposed limit on the maximum number of concurrently running tasks per job.

If this option or a corresponding value in qmon is specified, this value will be passed to defined JSV instances as a parameter with the name **tc**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-tcon y[es]|n[o]

Available for *qsub* only.

Can be used in conjunction with array jobs (see -t option) to submit a concurrent array job.

For a concurrent array job, either all tasks are started in one scheduling run or the whole job stays pending.

When combined with the **-now y** option, an immediate concurrent array job will be rejected if all tasks cannot be scheduled immediately.

The **-tcon y** switch cannot be combined with the **-tc** or the **-R** switch.

If this option is specified, its value (y or n) will be passed to defined JSV instances as a parameter with the name **tcon**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

Array task concurrency can be enabled and limited by using the MAX_TCON_TASKS qmaster_param setting in the global cluster configuration. See *sge_conf*(5). By default array task concurrency is disabled.

Submission of concurrent array jobs will be rejected when their size exceeds the settings of max_aj_tasks or max_aj_instances; see *sge_conf*(5). When max_aj_instances is lowered below the size of a pending concurrent array job, this job will stay pending.

-terse

Available for *qsub* only.

-terse causes *qsub* to display only the job-id of the job being submitted rather than the usual "Your job ..." string. In case of an error the error is reported on stderr as usual.

This can be helpful for scripts which need to parse *qsub* output to get the job-id.

Information that this switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

In Altair Grid Engine a variable named **terse** will be available and it will have the value **y** when the switch is used. This parameter can be changed in the JSV context to influence the behavior of the command-line client. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-umask parameter

With this option, the umask of a job and its output and error files can be set. The default is 0022. The value given here can only restrict the optional **execd_params** parameter JOB_UMASK or its default. See also *sge_conf*(5).

-u username,...

Available for *qalter* only. Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the **qalter -u '*'** command to modify all jobs of all users.

If you use the **-u** switch it is not permitted to specify an additional *wc_job_range_list*.

-v variable[=value],...

Available for *qsub*, *qrsh*, *qlogin* and *qalter*.

Defines or redefines the environment variables to be exported to the execution context of the job. If the **-v** option is present Altair Grid Engine will add the environment variables defined as arguments to the switch and optionally, values of specified variables, to the execution context of the job.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

All environment variables that are specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will be optionally fetched by the defined JSV instances during the job submission verification.

Information that the **-V** switch was specified during submission is not available in the JSV context of the open source version of Grid Engine.

(See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-verbose

Available only for *qrsh* and *qmake*(1).

Unlike *qsh* and *qlogin*, *qrsh* does not output any informational messages while establishing the session; it is compliant with the standard *rsh*(1) and *rlogin*(1) system calls. If the option **-verbose** is set, *qrsh* behaves like the *qsh* and *qlogin* commands, printing information about the process of establishing the *rsh*(1) or *rlogin*(1) session.

-verify

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Instead of submitting a job, prints detailed information about the would-be job as though *qstat*(1) -j were used, including the effects of command-line parameters and the external environment.

-V

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*.

Specifies that all environment variables active within the *qsub* utility be exported to the context of the job.

All environment variables specified with **-v**, **-V**, or the DISPLAY variable specified via **-display**, will optionally be fetched by the defined JSV instances during the job submission verification.

In Altair Grid Engine a variable named **-V** will be available and it will have the value **y** when the switch is used. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-w e|w|n|p|v

e|w|n|v are available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter*. **p** is also available for the listed commands except for *qalter*, where the functionality is deprecated.

Specifies the validation level applied to the job (to be submitted via_qsub_, *qlogin*, *qsh*, or *qrsh*) or the specified queued job (to be altered via *qalter*). The information displayed indicates whether the job can possibly be scheduled. Resource requests exceeding the amount of available resources cause jobs to fail the validation process.

The specifiers e, w, n and v define the following validation modes:

'n' none (default)
Switches off validation
'e' error
The validation process assumes an empty cluster without load values.
If the job can in principle run in this system, the validation
process will report success. Jobs to be submitted will be accepted
by the system, otherwise a validation report will be shown.

'w' warning
Same as 'e' with the difference that jobs to be submitted will be accepted by the system even if validation fails, and a validation report will be shown.
'v' verify
Same as 'e' with the difference that jobs to be submitted will not be submitted. Instead a validation report will be shown.
'p' poke
The validation step assumes the cluster as it is, with all resource utilization and load values in place. Jobs to be submitted will not be submitted even if validation is successful; instead a validation report will be shown.

e, **w** and **v** do not consider load values as part of the verification since they are assumed to be to volatile. Managers can change this behavior by defining the qmaster_param **CONSIDER_LOAD_DURING_VERIFY** which omits the necessity to define the maximum capacity in the complex_values for a resource on global, host, or queue level, but also causes the validation step to fail if the requested amount of resources exceeds the available amount reported as the load value at the current point in time.

Independent of **CONSIDER_LOAD_DURING_VERIFY**, setting the validation process will always use the maximum capacity of a resource if it is defined and if a load value for this resource is reported.

Note that the necessary checks are performance-consuming and hence the checking is switched off by default.

Please also note that enabled verifications are done during submission after JSV verification and adjustments have been applied. To enable requested verification before JSVs are handled, administrators have to define ENABLE_JOB_VERIFY_BEFORE_JSV as qmaster_param in the global configuration.

-wd working_dir

Available for *qsub*, *qsh*, *qrsh* and *qalter* only.

Execute the job from the directory specified in working_dir. This switch will activate the sge path aliasing facility, if the corresponding configuration files are present (see *sge_aliases*(5)).

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however. The parameter value will be available in defined JSV instances as a parameter with the name **cwd** (See the **-cwd** switch above or find more information concerning JSV in *sge_jsv*(5).)

-when [now|on_reschedule]

Available for *qalter* only.

qalter allows alteration of resource requests of running jobs. If -when now is set, the changes will be done immediately if possible (only for consumables). If -when on_reschedule (default) is set, the changes take effect when the job gets re-scheduled.

command

Available for *qsub* and *qrsh* only.

Specifies the job's scriptfile or binary. If not present or if the operand is the single-character string '-', *qsub* reads the script from standard input.

The command will be available in defined JSV instances as a parameter with the name **CMD-NAME**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-xd docker_option

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only when submitting Docker jobs.

Use the **-xd** switch for specifying arbitrary **docker run** options to be used in the creation of containers for Docker jobs. **docker run** means the **run** option of the **docker** command that is part of the Docker Engine.
If a **docker run** option and/or its arguments contain spaces, quoting is required, e.g. qsub -xd "-v /tmp:/hosts_tmp". Multiple **docker run** options can be specified as a comma-separated list with one **-xd** option, e.g. qsub -xd –net=usernet,–ip=192.168.99.10,–hostname=test.

-xd –help prints a list of **docker run** options, whether each is supported by Altair Grid Engine and if not supported, a comment describing why the option is not supported, which option to use instead, and how the **docker run** option is passed via the docker API to docker.

Placeholders can be used in arguments to Docker options. These placeholders are resolved with values the Altair Grid Engine Scheduler selected for the job based on the resource map (RSMAP) requests of the job that correspond to the placeholders.
These placeholders have the format:

<placeholder> := ${ <complex_name> "(" <index> ")" }

complex_name is defined in *sge_types*(1) and is the name of the resource map which is requested for this job.
index is the index of the element of the resource map to use, and the first element has index 0.

Using these placeholders is supported only if the RSMAP is of type "consumable=HOST"; "consumable=YES" and "consumable=JOB" are not supported, because the list of resource map elements granted for the parallel tasks on a certain host depend on the number of tasks scheduled there.

The substitution is equal for all PE tasks running on the same host, so likely it makes sense only to use the placeholder substitution in conjunction with PE allocation rule=1. If there is a "-masterl" request for that RSMAP, this request is valid for the whole master host anyway.

E.g.: If a resource map defines these values on a host: gpu_map=4(0 1 2 3), and this qsub command line is used:

qsub -l docker,docker_images="*some_image*",gpu_map=2
-xd "–device=/dev/gpu${gpu_map(0)}:/dev/gpu0,
–device=/dev/gpu${gpu_map(1)}:/dev/gpu1" …

and the scheduler selects the elements “1” and “3” from the resource map, this resuls in the command line being resolved to:

qsub -l docker,docker_images“*some_image*”,gpu_map=2
-xd “–device=/dev/gpu1:/dev/gpu0,
–device=/dev/gpu3:/dev/gpu1”…

which means the physical GPUs “gpu1” and “gpu3” are mapped to the virtual GPUs “gpu0” and “gpu1” inside the container and at the same time are exclusively reserved for the current job among all Altair Grid Engine jobs.

-xd “–group-add” and the special keyword SGE_SUP_GRP_EVAL

Using the special keyword **SGE_SUP_GRP_EVAL** with the **–group-add** option of the **-xd** switch allows automatic addition of all supplementary groups to the group list of the job user inside the Docker container. Additional group IDs can be specified by using the **–group-add** option multiple times.

E.g.:

qsub -l docker,docker_images=“*some_image*”, -xd “–group-add SGE_SUP_GRP_EVAL,–group-add 789” …

makes Altair Grid Engine add all additional group IDs of the job owner **on the execution host** as well as the group ID 789.

-xdv docker_volume

Available for *qsub*, *qsh*, *qrsh*, *qlogin* and *qalter* only.

When a job is running within a Docker container the **-xdv** switch can be used to specify docker volumes to be mounted into the docker container. **docker_volume** is specified following the syntax of the docker run command-line option -v; see the *docker run*(1) man page. Multiple volumes can be mounted by passing a comma-separated list of volumes to the **-xdv** switch or by repeating the **-xdv** switch.

The **-xdv** switch is deprecated and will be removed in future versions of Altair Grid Engine; use **-xd –volume** instead.

-xd_run_as_image_user y[es]|n[o]

Available for *qsub* and *qalter* only.

This option is available only if the **qmaster_params ENABLE_XD_RUN_AS_IMAGE_USER** is defined and set to **1** or **true**. This option is valid only for autostart Docker jobs, i.e. for Docker jobs that use the keyword **NONE** as the job to start. If this option is specified and set to **y** or **yes**, the autostart Docker job is started as the user defined in the Docker image from which the Docker container is created. If there is no user defined in the Docker image, the behavior is undefined. If this option is omitted or is set to **n** or **no**, the autostart Docker job is started as the user who submitted the job.

command_args

Available for *qsub*, *qrsh* and *qalter* only.

Arguments to the job. Not valid if the script is entered from standard input.

qalter allows changing this option even while the job executes. The modified parameter will be in effect only after a restart or migration of the job, however.

The number of command arguments is provided to configured JSV instances as a parameter with the name **CMDARGS**. In addition, the argument values can be accessed. Argument names have the format **CMDARG<number>** where **<number>** is a integer between 0 and **CMDARGS** - 1. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

xterm_args

Available for *qsh* only.

Arguments to the *xterm*(1) executable, as defined in the configuration. For details, refer to *sge_conf*(5).

Information concerning **xterm_args** will be available in JSV context as parameters with the names **CMDARGS** and **CMDARG<number>**. Find more information above in section **command_args**. (See the **-jsv** option above or find more information concerning JSV in *sge_jsv*(5).)

-suspend_remote y[es]|n[o]

Available for *qrsh* only. Suspend qrsh client as also the process on execution host.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *qsub*, *qsh*, *qlogin* or *qalter* use (in the following order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition, the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will instead use a services map entry for the service "sge_qmaster" to define that port.

DISPLAY

For *qsh* jobs the DISPLAY has to be specified at job submission. If the DISPLAY is not set via **-display** or the **-v** switch, the contents of the DISPLAY environment variable are used.

In addition to those environment variables specified to be exported to the job via the **-v** or the **-V** options, (see above) *qsub*, *qsh*, and *qlogin* add the following variables with the indicated values to the variable list:

SGE_O_HOME

the home directory of the submitting client.

SGE_O_HOST

the name of the host on which the submitting client is running.

SGE_O_LOGNAME

the LOGNAME of the submitting client.

SGE_O_MAIL

the MAIL of the submitting client. This is the mail directory of the submitting client.

SGE_O_PATH

the executable search path of the submitting client.

SGE_O_SHELL

the SHELL of the submitting client.

SGE_O_TZ

the time zone of the submitting client.

SGE_O_WORKDIR

the absolute path of the current working directory of the submitting client.

Furthermore, Altair Grid Engine sets additional variables in the job's environment, as listed below:

ARC

SGE_ARCH

The Altair Grid Engine architecture name of the node on which the job is running. The name is compiled into the *sge_execd*(8) binary.

SGE_BINDING

This variable contains the selected operating system internal processor numbers. They might be more than selected cores in the presence of SMT or CMT because each core could be represented by multiple processor identifiers. The processor numbers are space-separated.

SGE_CKPT_ENV

Specifies the checkpointing environment (as selected with the **-ckpt** option) under which a checkpointing job executes. Only set for checkpointing jobs.

SGE_CKPT_DIR

Only set for checkpointing jobs. Contains path *ckpt_dir* (see *sge_checkpoint*(5)) of the checkpoint interface.

SGE_CWD_PATH

Specifies the current working directory where the job was started.

SGE_STDERR_PATH

The pathname of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDOUT_PATH

The pathname of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts.

SGE_STDIN_PATH

The pathname of the file from which the standard input stream of the job is taken. This variable might be used in combination with SGE_O_HOST in prolog/epilog scripts to transfer the input file from the submit to the execution host.

SGE_JOB_SPOOL_DIR

The directory used by *sge_shepherd*(8) to store job-related data during job execution. This directory is owned by root or by a Altair Grid Engine administrative account and commonly is not open for read or write access by regular users.

SGE_TASK_ID

The index number of the current array job task (see **-t** option above). This is a unique number in each array job and can be used to reference different input data records, for example. This environment variable is set to "undefined" for non-array jobs. It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_FIRST

The index number of the first array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_LAST

The index number of the last array job task (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

SGE_TASK_STEPSIZE

The step size of the array job specification (see **-t** option above). It is possible to change the predefined value of this variable with **-v** or **-V** (see options above).

ENVIRONMENT

The ENVIRONMENT variable is set to BATCH to identify that the job is being executed under Altair Grid Engine control.

HOME

The user's home directory path from the *passwd*(5) file.

HOSTNAME

The hostname of the node on which the job is running.

JOB_ID

A unique identifier assigned by the *sge_qmaster*(8) when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.

JOB_NAME

The job name. For batch jobs or jobs submitted by *qrsh* with a command, the job name is built using as its basename the *qsub* script filename or the *qrsh* command. For interactive jobs it is set to 'INTERACTIVE' for *qsh* jobs, 'QLOGIN' for *qlogin* jobs, and 'QRLOGIN' for *qrsh* jobs without a command.

This default may be overwritten by the *-N* option.

JOB_SCRIPT

The path to the job script which is executed. The value cannot be overwritten by the **-v** or **-V** option.

LOGNAME

The user's login name from the *passwd*(5) file.

NHOSTS

The number of hosts in use by a parallel job.

NQUEUES

The number of queues allocated for the job (always 1 for serial jobs).

NSLOTS

The number of queue slots in use by a parallel job.

PATH

A default shell search path of:

/usr/local/bin:/usr/ucb:/bin:/usr/bin

SGE_BINARY_PATH

The path where the Altair Grid Engine binaries are installed. The value is the concatenation of the cluster configuration value **binary_path** and the architecture name **$SGE_ARCH** environment variable.

PE

The parallel environment under which the job executes (for parallel jobs only).

PE_HOSTFILE

The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Altair Grid Engine. See the description of the **$pe_hostfile** parameter in sge_pe*(5)* for details on the format of this file. The environment variable is only available for parallel jobs.

QUEUE

The name of the cluster queue in which the job is running.

REQUEST

Available for batch jobs only.

The request name of a job as specified with the **-N** switch (see above) or taken as the name of the job script file.

RESTARTED

This variable is set to 1 if a job was restarted either after a system crash or after a migration for a checkpointing job. The variable has the value 0 otherwise.

SHELL

The user's login shell from the *passwd*(5) file. **Note:** This is not necessarily the shell in use for the job.

TMPDIR

The absolute path to the job's temporary working directory.

TMP

The same as TMPDIR; provided for compatibility with NQS.

TZ

The time zone variable imported from *sge_execd*(8) if set.

USER

The user's login name from the *passwd*(5) file.

SGE_JSV_TIMEOUT

If the response time of the client JSV is greater than this timeout value, the JSV will attempt to be re-started. The default value is 10 seconds. The value must be greater than 0. If the timeout has been reached, the JSV will only try to re-start once; if the timeout is reached again an error will occur.

SGE_JOB_EXIT_STATUS

This value contains the exit status of the job script itself. This is the same value that can later be found in the **exit_status** field in the **qacct -j <job_id>** output. This variable is available in the pe_stop and epilog environment only.

SGE_RERUN_REQUESTED

This value indicates whether a job rerun on error was explicitly requested. This value is 0 if the **-r** option was not specified on the job submit command line, by the job class, or by a JSV script; the value is 1 if **-r y** was requested; the value is 2 if **-r n** was requested. For interactive jobs submitted by **qrsh** or **qlogin**, **-r n** is always implicitly requested, and therefore the value of this environment variable is always 2. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

SGE_RERUN_JOB

This value indicates whether the job is going to be rescheduled on error. This value is 0 if the job will not be rerun and 1 if it will be rerun on error. To determine this value, the explicitly requested (or for interactive jobs, implicitly requested) **-r** option is used. If this is option is not specified, the queue configuration value **rerun** is used as the default value. This variable is available in the prolog, pe_start, job, pe_stop, and epilog environment.

AGE_MISTRAL

This is set to override the Mistral profiling behaviour in combination with AGE_MISTRAL_MODE execd_params config value.

AGE_BREEZE

This is set to override the Breeze profiling behaviour in combination with AGE_BREEZE_MODE execd_params config value.

When set (1 or true) or unset (0 or false) or not expilicitly set(-), Breeze/Mistral profiling will happen based on the execd_params values as following:

AGE_BREEZE/AGE_MISTRAL	execd_params value	Result
- 0 1	ALWAYS	1 1 1
- 0 1	NEVER	0 0 0
- 0 1	IF_REQUESTED	0 0 1
- 0 1	DEFAULT_ON	1 0 1

RESTRICTIONS

There is no controlling terminal for batch jobs under Altair Grid Engine, and any tests or actions on a controlling terminal will fail. If these operations are in your .login or .cshrc file, they may cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your .login:

if ($?JOB_NAME) then

echo “Altair Grid Engine spooled job”
exit 0

endif

Don’t forget to set your shell’s search path in your shell start-up before this code.

EXIT STATUS

The following exit values are returned:

0

Operation was executed successfully.

25

It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in *sge_conf*(5).

0

Error occurred.

EXAMPLES

The following is the simplest form of a Altair Grid Engine script file.

```
==========================================================

#!/bin/csh
a.out

==========================================================
```

The next example is a more complex Altair Grid Engine script.

```
==========================================================

#!/bin/csh

# Which account to be charged CPU time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@northpole,claus@northpole

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out

==========================================================
```

FILES

$REQUEST.oJID[.TASKID] STDOUT of job #JID
$REQUEST.eJID[.TASKID] STDERR of job
$REQUEST.poJID[.TASKID] STDOUT of par. env. of job
$REQUEST.peJID[.TASKID] STDERR of par. env. of job

$cwd/.sge_aliases cwd path aliases
$cwd/.sge_request cwd default request
$HOME/.sge_aliases user path aliases
$HOME/.sge_request user default request
<sge_root>/<cell>/common/sge_aliases
cluster path aliases
<sge_root>/<cell>/common/sge_request cluster default request
<sge_root>/<cell>/common/act_qmaster Altair Grid Engine master host file

SEE ALSO

sge_intro(1), *qconf(1)*, *qdel(1)*, *qhold(1)*, *qmod(1)*, *qrls(1)*, *qstat(1)*, *sge_accounting(5)*, *sge_session_conf(5)*, *sge_aliases(5)*, *sge_conf(5)*, *sge_job_class(5)*, *sge_request(5)*, *sge_types(1)*, *sge_pe(5)*, *sge_resource_map(5)*, *sge_complex(5)*.

COPYRIGHT

drmaa_allocate_job

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_attributes

NAME

drmaa_get_attribute_names, drmaa_get_vector_attribute_names, drmaa_get_next_attr_name, drmaa_get_num_attr_names, drmaa_release_attr_names - DRMAA job template attributes

SYNOPSIS

```
#include "drmaa.h"

int drmaa_get_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_name(
        drmaa_attr_names_t* values,
        char *value,
        int value_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_names(
        drmaa_attr_names_t* values,
        int *size
);

void drmaa_release_attr_names(
        drmaa_attr_names_t* values
);
```

DESCRIPTION

The drmaa_get_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported non-vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The number of names in the names string vector can be determined using *drmaa_get_num_attr_names*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_attribute*(3) and *drmaa_get_attribute*(3) for setting and inspecting non-vector attributes.

drmaa_get_vector_attribute_names()

The drmaa_get_vector_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_vector_attribute*(3) and *drmaa_get_vector_attribute*(3) for setting and inspecting vector attributes.

drmaa_get_next_attr_name()

Each time drmaa_get_next_attr_name() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA names string vector, *values*. Once the names list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_names()

The drmaa_get_num_attr_names() returns into *size* the number of entries in the DRMAA names string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_names()

The drmaa_release_attr_names() function releases all resources associated with the DRMAA names string vector, *values*.

Attribute Priorities

DRMAA job template attributes can be set from six different sources. In order of precedence, from lowest to highest, these are: options set by DRMAA automatically by default, options set in the *sge_request*(5) file(s), options set in the script file, options set by the drmaa_job_category attribute, options set by the drmaa_native_specification attribute, and options set through other DRMAA attributes.

By default DRMAA sets four options for all jobs. These are “-p 0”, “-b yes”, “-shell no”, and “-w e”. This means that by default, all jobs will have priority 0, all jobs will be treated as binary, i.e. no scripts args will be parsed, all jobs will be executed without a wrapper shell, and jobs which are unschedulable will cause a submit error.

The *sge_request*(5) file, found in the $SGE_ROOT/$SGE_CELL/common directory, may contain options to be applied to all jobs. The .sge_request file found in the user’s home directory or the current working directory may also contain options to be applied to certain jobs. See *sge_request*(5) for more information. If the *sge_request*(5) file contains “-b no” or if the drmaa_native_specification attribute is set and contains “-b no”, the script file will be parsed for in-line arguments. Otherwise, no scripts args will be interpreted. See *qsub*(1) for more information. If the drmaa_job_category attribute is set, and the category it points to exists in one of the *qtask* files, the options associated with that category will be applied to the job template.

If the drmaa_native_specification attribute is set, all options contained therein will be applied to the job template. See the drmaa_native_specification below for more information. Other DRMAA attributes will override any previous settings. For example, if the sge_request file contains “-j y”, but the drmaa_join_files attribute is set to “n”, the ultimate result is that the input and output files will remain separate.

For various reasons, some options are silently ignored by DRMAA. Setting any of these options will have no effect. The ignored options are: -cwd, -help, -sync, -t, -verify, -w w, and -w v. The -cwd option can be re-enabled by setting the environment variable, SGE_DRMAA_ALLOW_CWD. However, the -cwd option is not thread safe and should not be used in a multi-threaded context.

Attribute Correlations

The following DRMAA attributes correspond to the following *qsub*(1) options:

```
DRMAA Attribute                  qsub Option
-----------------------------------------------------
drmaa_remote_command             script file
drmaa_v_argv                     script file args
drmaa_js_state = "drmaa_hold"    -h
drmaa_v_env                      -v
drmaa_wd = \$PWD                  -cwd
drmaa_job_category               (qtsch qtask)*
drmaa_native_specification       ALL*
drmaa_v_email                    -M
drmaa_block_email = "1"          -m n
drmaa_start_time                 -a
drmaa_job_name                   -N
drmaa_input_path                 -i
drmaa_output_path                -o
drmaa_error_path                 -e
drmaa_join_files                 -j
drmaa_transfer_files             (prolog and epilog)*
```

```
* See the individual attribute description below
```

DRMAA JOB TEMPLATE ATTRIBUTES

drmaa_remote_command - "<remote_command>"

Specifies the remote command to execute. The *remote_command* must be the path of an executable that is available at the job's execution host. If the path is relative, it is assumed to be relative to the working directory, usually set through the drmaa_wd attribute. If working directory is not set, the path is assumed to be relative to the user's home directory.

The file pointed to by remote_command may either be an executable binary or an executable script. If a script, it must include the path to the shell in a #! line at the beginning of the script. By default, the remote command will be executed directly, as by *exec*(2). To have the remote command executed in a shell, such as to preserve environment settings, use the drmaa_native_specification attribute to include the "-shell yes" option. Jobs which are executed by a wrapper shell fail differently from jobs which are executed directly. When a job which contains a user error, such as an invalid path to the executable, is executed by a wrapper shell, the job will execute successfully, but exit with a return code of 1. When a job which contains such an error is executed directly, it will enter the DRMAA_PS_FAILED state upon execution.

drmaa_js_state - "{drmaa_hold|drmaa_active}"

Specifies the job state at submission. The string values 'drmaa_hold' and 'drmaa_active' are supported. When 'drmaa_active' is used the job is submitted in a runnable state. When 'drmaa_hold' is used the job is submitted in user hold state (either DRMAA_PS_USER_ON_HOLD or DRMAA_PS_USER_SYSTEM_ON_HOLD). This attribute is largely equivalent to the *qsub*(1) submit option '-h'.

drmaa_wd - "<directory_name>"

Specifies the directory name where the job will be executed. A '$drmaa_hd_ph$' placeholder at the beginning of the *directory_name* denotes the remaining string portion as a relative directory name that is resolved relative to the job user's home directory at the execution host. When the DRMAA job template is used for bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *directory_name* to cause a substitution with the parametric job's index. The *directory_name* must be specified in a syntax that is common at the host where the job is executed. If set to a relative path and no placeholder is used, a path relative to the user's home directory is assumed. If not set, the working directory will default to the user's home directory. If set and the given directory does not exist the job will enter the DRMAA_PS_FAILED state when run.

Note that the working directory path is the path on the execution host. If binary mode is disabled, an attempt to find the job script will be made, relative to the working directory

path. That means that the path to the script must be the same on both the submission and execution hosts.

drmaa_job_name - "<job_name>"

Specifies the job's name. Setting the job name is equivalent to use of *qsub*(1) submit option '-N' with *job_name* as option argument.

drmaa_input_path - "[<hostname>]:<file_path>"

Specifies the standard input of the job. Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream. If the standard input is set it specifies the network path of the job's input stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'i', the input file will be fetched by Altair Grid Engine from the specified host or from the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for bulk job submission, (See also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be read the job enters the state DRMAA_PS_FAILED.

drmaa_output_path - "[<hostname>]:<file_path>"

Specifies the standard output of the job. If not explicitly set in the job template, the whereabouts of the job's output stream is not defined. If set, this attribute specifies the network path of the job's output stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'o', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the

file_path denotes the remaining portion of *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.

drmaa_error_path - "[<hostname>]:<file_path>"

Specifies the standard error of the job. If not explicitly set in the job template, the whereabouts of the job's error stream is not defined. If set, this attribute specifies the network path of the job's error stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'e', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED. The attribute name is drmaa_error_path.

drmaa_join_files - "{y|n}"

Specifies if the job's error stream should be intermixed with the output stream. If not explicitly set in the job template the attribute defaults to 'n'. Either 'y' or 'n' can be specified. If 'y' is specified Altair Grid Engine will ignore the value of the 'drmaa_error_path' job template attribute and intermix the standard error stream with the standard output stream as specified with 'drmaa_output_path'.

drmaa_v_argv - "argv1 argv2 . . ."

Specifies the arguments to the job.

drmaa_job_category - "<category>"

Specifies the DRMAA job category. The *category* string is used by Altair Grid Engine as a reference into the *qtask* file. Certain *qsub*(1) options used in the referenced qtask file line are applied to the job template before submission to allow site-specific resolving of resources and/or policies. The cluster qtask file, the local qtask file, and the user qtask file

are searched. Job settings resulting from job template category are overridden by settings resulting from the job template drmaa_native_specification attribute as well as by explicit DRMAA job template settings.

In order to avoid collisions with command names in the qtask files, it is recommended that DRMAA job category names take the form: <category_name>.cat.

The options -help, -sync, -t, -verify, and -w w|v are ignored. The -cwd option is ignored unless the $SGE_DRMAA_ALLOW_CWD environment variable is set.

drmaa_native_specification - "<native_specification>"

Specifies Altair Grid Engine native *qsub*(1) options which will be interpreted as part of the DRMAA job template. All options available to *qsub*(1) command may be used in the *native_specification*, except for -help, -sync, -t, -verify, and -w w|v. The -cwd option may only be used if the SGE_DRMAA_ALLOW_CWD environment variable is set. This is because the current parsing algorithm for -cwd is not thread-safe. Options set in the *native specification* will be overridden by the corresponding DRMAA attributes. See *qsub*(1) for more information on qsub options.

drmaa_v_env - "<name1>=<value1> <name2>=<value2> ..."

Specifies the job environment. Each environment *value* defines the remote environment. The *value* overrides the remote environment values if there is a collision.

drmaa_v_email - "<email1> <email2> ..."

Specifies e-mail addresses that are used to report the job completion and status.

drmaa_block_email - "{0|1}"

Specifies whether e-mail sending shall blocked or not. By default email is not sent. If, however, a setting in a cluster or user settings file or the 'drmaa_native_specification' or 'drmaa_job_category' attribute enables sending email in association with job events, the 'drmaa_block_email' attribute will override that setting, causing no email to be sent.

drmaa_start_time - "[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]"

Specifies the earliest time when the job may be eligible to be run where

```
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
```

```
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
```

If the optional UTC-offset is not specified, the offset associated with the local timezone will be used. If the day (DD) is not specified, the current day will be used unless the specified hour:mm:ss has already elapsed, in which case the next day will be used. Similarly for month (MM), year (YY), and century-1 (CC). Example: The time: Sep 3 4:47:27 PM PDT 2002, could be represented as: 2002/09/03 16:47:27 -07:00.

drmaa_transfer_files - "[i][o][e]"

Specifies, which of the standard I/O files (stdin, stdout and stderr) are to be transferred to/from the execution host. If not set, defaults to "". Any combination of 'e', 'i' and 'o' may be specified. See drmaa_input_path, drmaa_output_path and drmaa_error_path for information about how to specify the standard input file, standard output file and standard error file. The file transfer mechanism itself must be configured by the administrator (see *sge_conf*(5)). When it is configured, the administrator has to enable drmaa_transfer_files. If it is not configured,"drmaa_transfer_files" is not enabled and can't be used.

ENVIRONMENTAL VARIABLES

- *SGE_DRMAA_ALLOW_CWD* Enables the parsing of the -cwd option from the sge_request file(s), job category, and/or the native specification attribute. This option is disabled by default because the algorithm for parsing the -cwd option is not thread-safe.
- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_next_attr_name() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_jobtemplate(3) and *drmaa_submit*(3).

COPYRIGHT

drmaa_control

NAME

drmaa_job_ps, drmaa_control, - Monitor and control jobs

SYNOPSIS

```
#include "drmaa.h"

int drmaa_job_ps(
        const char *job_id,
        int *remote_ps,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_control(
        const char *jobid,
        int action,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_job_ps() function returns the status of the Altair Grid Engine job *job_id* into the integer pointed to by *remote_ps*. Possible return values are

```
DRMAA_PS_UNDETERMINED         job status cannot be determined
DRMAA_PS_QUEUED_ACTIVE        job is queued and active
DRMAA_PS_SYSTEM_ON_HOLD       job is queued and in system hold
DRMAA_PS_USER_ON_HOLD         job is queued and in user hold
DRMAA_PS_USER_SYSTEM_ON_HOLD job is queued and in user and system hold
DRMAA_PS_RUNNING              job is running
DRMAA_PS_SYSTEM_SUSPENDED     job is system suspended
DRMAA_PS_USER_SUSPENDED       job is user suspended
DRMAA_PS_DONE                 job finished normally
DRMAA_PS_FAILED               job finished, but failed
```

Jobs' user hold and user suspend states can be controlled via *drmaa_control(3)*. For affecting system hold and system suspend states the appropriate Altair Grid Engine interfaces must be used.

drmaa_control()

The drmaa_control() function applies control operations on Altair Grid Engine jobs. *jobid* may contain either an Altair Grid Engine jobid or 'DRMAA_JOB_IDS_SESSION_ALL' to refer to all jobs submitted during the DRMAA session opened using drmaa_init(3). Legal values for *action* and their meanings are:

```
DRMAA_CONTROL_SUSPEND        suspend the job
DRMAA_CONTROL_RESUME         resume the job
DRMAA_CONTROL_HOLD           put the job on-hold
DRMAA_CONTROL_RELEASE        release the hold on the job
DRMAA_CONTROL_TERMINATE      kill the job
```

The DRMAA suspend/resume operations are equivalent to the use of `-sj <jobid>' and -usj` ' options with Altair Grid Engine qmod(1). The DRMAA hold/release operations are equivalent to the use of Altair Grid Engine qhold(1) and qrls(1). The DRMAA terminate operation is equivalent to the use of Altair Grid Engine qdel(1). Only user hold and user suspend can be controlled via drmaa_control(3). For affecting system hold and system suspend states the appropriate Altair Grid Engine interfaces must be used.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster(8)* is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_job_ps(), and drmaa_control() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer, *error_diagnosis*.

ERRORS

The drmaa_job_ps(), and drmaa_control() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request was not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

DRMAA_ERRNO_INVALID_JOB

The specified job does not exist.
The drmaa_control() will fail if:

DRMAA_ERRNO_RESUME_INCONSISTENT_STATE

The job is not suspended. The resume request will not be processed.

DRMAA_ERRNO_SUSPEND_INCONSISTENT_STATE

The job is not running and thus cannot be suspended.

DRMAA_ERRNO_HOLD_INCONSISTENT_STATE

The job cannot be moved to a hold state.

DRMAA_ERRNO_RELEASE_INCONSISTENT_STATE

The job is not in a hold state.

SEE ALSO

drmaa_submit(3) drmaa_wait(3).

COPYRIGHT

drmaa_delete_job

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_exit

NAME

drmaa_init, drmaa_exit - Start/finish Altair Grid Engine DRMAA session

SYNOPSIS

```
#include "drmaa.h"

int drmaa_init(
        const char *contact,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_exit(
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

drmaa_init()

The drmaa_init() function initializes the Altair Grid Engine DRMAA API library for all threads of the process and creates a new DRMAA session. This routine must be called once before any other DRMAA call, except for *drmaa_version(3)*, *drmaa_get_DRM_system(3)*, and *drmaa_get_DRMAA_implementation(3)*. Except for the above listed functions, no DRMAA functions may be called before the drmaa_init() function *completes*. Any DRMAA function which is called before the drmaa_init() function completes will return a DRMAA_ERRNO_NO_ACTIVE_SESSION error. *Contact* is an implementation dependent string which may be used to specify which Altair Grid Engine cell to use. If *contact* is NULL, the default Altair Grid Engine cell will be used. In the 1.0 implementation *contact* may have the following value: *session=<session_id>*. To determine the session_id, the *drmaa_get_contact(3)* function should be called after the session has already been initialized. By passing the *session=<session_id>* string to the drmaa_init() function, instead of creating a new session, DRMAA will attempt to reconnect to the session indicated by the *session_id*. The result of reconnecting to a previous session is that all jobs previously submitted in that session *that_are_still_running* will be available in the DRMAA session. Note, however,that jobs which ended before the call to drmaa_init() may not be available or may have no associated exit information or resource usage data.

drmaa_exit()

The drmaa_exit() function closes the DRMAA session for all threads and must be called before process termination. The drmaa_exit() function may be called only once by a single thread in the process and may only be called after the drmaa_init() function has completed. Any DRMAA function, other than *drmaa_init(3)*,which is called after the drmaa_exit() function completes will return a DRMAA_ERRNO_NO_ACTIVE_SESSION error.

The drmaa_exit() function does necessary clean up of the DRMAA session state, including unregistering from the sge_qmaster(8). If the drmaa_exit() function is not called, the qmaster will store events for the DRMAA client until the connection times out, causing extra work for the qmaster and consuming system resources.

Submitted jobs are not affected by the drmaa_exit() function.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

he name of the default cell, i.e. *default*.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which the sge_qmaster is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_init() and drmaa_exit() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_init() and drmaa_exit() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.
The drmaa_init() will fail if:

DRMAA_ERRNO_INVALID_CONTACT_STRING

Initialization failed due to invalid contact string.

DRMAA_ERRNO_DEFAULT_CONTACT_STRING_ERROR

Could not use the default contact string to connect to DRM system.

DRMAA_ERRNO_DRMS_INIT_FAILED

Initialization failed due to failure to init DRM system.

DRMAA_ERRNO_ALREADY_ACTIVE_SESSION

Initialization failed due to existing DRMAA session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_exit() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_DRMS_EXIT_ERROR

DRM system disengagement failed.

SEE ALSO

drmaa_submit(3).

COPYRIGHT

drmaa_get_DRMAA_implementation

NAME

drmaa_strerror, drmaa_get_contact, drmaa_version, drmaa_get_DRM_system - Miscellaneous DRMAA functions.

SYNOPSIS

```
#include "drmaa.h"

const char *drmaa_strerror(
        int drmaa_errno
);

int drmaa_get_contact(
        char *contact,
        size_t contact_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_version(
        unsigned int *major,
        unsigned int *minor,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRM_system(
        char *drm_system,
        size_t drm_system_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRMAA_implementation(
        char *drm_impl,
        size_t drm_impl_len,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_strerror() function returns a message text associated with the DRMAA error number,*drmaa_errno*. For invalid DRMAA error codes 'NULL' is returned.

drmaa_get_contact()

The drmaa_get_contact() returns an opaque string containing contact information related to the current DRMAA session to be used with the *drmaa_init*(3) function. The opaque string contains the information required by drmaa_init() to reconnect to the current session instead of creating a new session. *drmaa_init*(3) function.

The drmaa_get_contact() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_version()

The drmaa_version() function returns into the integers pointed to by *major* and *minor*, the major and minor version numbers of the DRMAA library. For a DRMAA 1.0 compliant implementation '1' and '0' will be returned in *major* and *minor*, respectively.

drmaa_get_DRM_system()

The drmaa_get_DRM_system() function returns into *drm_system* up to *drm_system_len* characters of a string containing Altair Grid Engine product and version information.

The drmaa_get_DRM_system() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_get_DRMAA_implementation()

The drmaa_get_DRMAA_implementation() function returns into *drm_system* up to *drm_system_len* characters of a string containing the Altair Grid Engine DRMAA implementation version information. In the current implementation, the drmaa_get_DRMAA_implementation() function returns the same result as the drmaa_get_DRM_system() function.

The drmaa_get_DRMAA_implementation() function returns the same value before and after *drmaa_init*(3) is called.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_contact(), drmaa_version(), and drmaa_get_DRM_system() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_contact(), drmaa_version(), drmaa_get_DRM_system(), and drmaa_get_DRMAA_implementation will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_contact() and drmaa_get_DRM_system() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

SEE ALSO

drmaa_session(3).

COPYRIGHT

drmaa_get_DRM_system

NAME

drmaa_strerror, drmaa_get_contact, drmaa_version, drmaa_get_DRM_system - Miscellaneous DRMAA functions.

SYNOPSIS

```
#include "drmaa.h"

const char *drmaa_strerror(
        int drmaa_errno
);

int drmaa_get_contact(
        char *contact,
        size_t contact_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_version(
        unsigned int *major,
        unsigned int *minor,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRM_system(
        char *drm_system,
        size_t drm_system_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRMAA_implementation(
        char *drm_impl,
        size_t drm_impl_len,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_strerror() function returns a message text associated with the DRMAA error number,*drmaa_errno*. For invalid DRMAA error codes 'NULL' is returned.

drmaa_get_contact()

The drmaa_get_contact() returns an opaque string containing contact information related to the current DRMAA session to be used with the *drmaa_init*(3) function. The opaque string contains the information required by drmaa_init() to reconnect to the current session instead of creating a new session. *drmaa_init*(3) function.

The drmaa_get_contact() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_version()

The drmaa_version() function returns into the integers pointed to by *major* and *minor*, the major and minor version numbers of the DRMAA library. For a DRMAA 1.0 compliant implementation '1' and '0' will be returned in *major* and *minor*, respectively.

drmaa_get_DRM_system()

The drmaa_get_DRM_system() function returns into *drm_system* up to *drm_system_len* characters of a string containing Altair Grid Engine product and version information.

The drmaa_get_DRM_system() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_get_DRMAA_implementation()

The drmaa_get_DRMAA_implementation() function returns into *drm_system* up to *drm_system_len* characters of a string containing the Altair Grid Engine DRMAA implementation version information. In the current implementation, the drmaa_get_DRMAA_implementation() function returns the same result as the drmaa_get_DRM_system() function.

The drmaa_get_DRMAA_implementation() function returns the same value before and after *drmaa_init*(3) is called.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_contact(), drmaa_version(), and drmaa_get_DRM_system() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_contact(), drmaa_version(), drmaa_get_DRM_system(), and drmaa_get_DRMAA_implementation will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_contact() and drmaa_get_DRM_system() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

SEE ALSO

drmaa_session(3).

COPYRIGHT

drmaa_get_attribute

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_get_attribute_names

NAME

drmaa_get_attribute_names, drmaa_get_vector_attribute_names, drmaa_get_next_attr_name, drmaa_get_num_attr_names, drmaa_release_attr_names - DRMAA job template attributes

SYNOPSIS

```
#include "drmaa.h"

int drmaa_get_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_name(
        drmaa_attr_names_t* values,
        char *value,
        int value_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_names(
        drmaa_attr_names_t* values,
        int *size
);

void drmaa_release_attr_names(
        drmaa_attr_names_t* values
);
```

DESCRIPTION

The drmaa_get_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported non-vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The number of names in the names string vector can be determined using *drmaa_get_num_attr_names*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_attribute*(3) and *drmaa_get_attribute*(3) for setting and inspecting non-vector attributes.

drmaa_get_vector_attribute_names()

The drmaa_get_vector_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_vector_attribute*(3) and *drmaa_get_vector_attribute*(3) for setting and inspecting vector attributes.

drmaa_get_next_attr_name()

Each time drmaa_get_next_attr_name() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA names string vector, *values*. Once the names list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_names()

The drmaa_get_num_attr_names() returns into *size* the number of entries in the DRMAA names string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_names()

The drmaa_release_attr_names() function releases all resources associated with the DRMAA names string vector, *values*.

Attribute Priorities

DRMAA job template attributes can be set from six different sources. In order of precedence, from lowest to highest, these are: options set by DRMAA automatically by default, options set in the *sge_request*(5) file(s), options set in the script file, options set by the drmaa_job_category attribute, options set by the drmaa_native_specification attribute, and options set through other DRMAA attributes.

By default DRMAA sets four options for all jobs. These are “-p 0”, “-b yes”, “-shell no”, and “-w e”. This means that by default, all jobs will have priority 0, all jobs will be treated as binary, i.e. no scripts args will be parsed, all jobs will be executed without a wrapper shell, and jobs which are unschedulable will cause a submit error.

The *sge_request*(5) file, found in the $SGE_ROOT/$SGE_CELL/common directory, may contain options to be applied to all jobs. The .sge_request file found in the user’s home directory or the current working directory may also contain options to be applied to certain jobs. See *sge_request*(5) for more information. If the *sge_request*(5) file contains “-b no” or if the drmaa_native_specification attribute is set and contains “-b no”, the script file will be parsed for in-line arguments. Otherwise, no scripts args will be interpreted. See *qsub*(1) for more information. If the drmaa_job_category attribute is set, and the category it points to exists in one of the *qtask* files, the options associated with that category will be applied to the job template.

If the drmaa_native_specification attribute is set, all options contained therein will be applied to the job template. See the drmaa_native_specification below for more information. Other DRMAA attributes will override any previous settings. For example, if the sge_request file contains “-j y”, but the drmaa_join_files attribute is set to “n”, the ultimate result is that the input and output files will remain separate.

For various reasons, some options are silently ignored by DRMAA. Setting any of these options will have no effect. The ignored options are: -cwd, -help, -sync, -t, -verify, -w w, and -w v. The -cwd option can be re-enabled by setting the environment variable, SGE_DRMAA_ALLOW_CWD. However, the -cwd option is not thread safe and should not be used in a multi-threaded context.

Attribute Correlations

The following DRMAA attributes correspond to the following *qsub*(1) options:

```
DRMAA Attribute                  qsub Option
-----------------------------------------------------
drmaa_remote_command             script file
drmaa_v_argv                     script file args
drmaa_js_state = "drmaa_hold"    -h
drmaa_v_env                      -v
drmaa_wd = \$PWD                  -cwd
drmaa_job_category               (qtsch qtask)*
drmaa_native_specification       ALL*
drmaa_v_email                    -M
drmaa_block_email = "1"          -m n
drmaa_start_time                 -a
drmaa_job_name                   -N
drmaa_input_path                 -i
drmaa_output_path                -o
drmaa_error_path                 -e
drmaa_join_files                 -j
drmaa_transfer_files             (prolog and epilog)*
```

```
* See the individual attribute description below
```

DRMAA JOB TEMPLATE ATTRIBUTES

drmaa_remote_command - "<remote_command>"

Specifies the remote command to execute. The *remote_command* must be the path of an executable that is available at the job's execution host. If the path is relative, it is assumed to be relative to the working directory, usually set through the drmaa_wd attribute. If working directory is not set, the path is assumed to be relative to the user's home directory.

The file pointed to by remote_command may either be an executable binary or an executable script. If a script, it must include the path to the shell in a #! line at the beginning of the script. By default, the remote command will be executed directly, as by *exec*(2). To have the remote command executed in a shell, such as to preserve environment settings, use the drmaa_native_specification attribute to include the "-shell yes" option. Jobs which are executed by a wrapper shell fail differently from jobs which are executed directly. When a job which contains a user error, such as an invalid path to the executable, is executed by a wrapper shell, the job will execute successfully, but exit with a return code of 1. When a job which contains such an error is executed directly, it will enter the DRMAA_PS_FAILED state upon execution.

drmaa_js_state - "{drmaa_hold|drmaa_active}"

Specifies the job state at submission. The string values 'drmaa_hold' and 'drmaa_active' are supported. When 'drmaa_active' is used the job is submitted in a runnable state. When 'drmaa_hold' is used the job is submitted in user hold state (either DRMAA_PS_USER_ON_HOLD or DRMAA_PS_USER_SYSTEM_ON_HOLD). This attribute is largely equivalent to the *qsub*(1) submit option '-h'.

drmaa_wd - "<directory_name>"

Specifies the directory name where the job will be executed. A '$drmaa_hd_ph$' placeholder at the beginning of the *directory_name* denotes the remaining string portion as a relative directory name that is resolved relative to the job user's home directory at the execution host. When the DRMAA job template is used for bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *directory_name* to cause a substitution with the parametric job's index. The *directory_name* must be specified in a syntax that is common at the host where the job is executed. If set to a relative path and no placeholder is used, a path relative to the user's home directory is assumed. If not set, the working directory will default to the user's home directory. If set and the given directory does not exist the job will enter the DRMAA_PS_FAILED state when run.

Note that the working directory path is the path on the execution host. If binary mode is disabled, an attempt to find the job script will be made, relative to the working directory

path. That means that the path to the script must be the same on both the submission and execution hosts.

drmaa_job_name - "<job_name>"

Specifies the job's name. Setting the job name is equivalent to use of *qsub*(1) submit option '-N' with *job_name* as option argument.

drmaa_input_path - "[<hostname>]:<file_path>"

Specifies the standard input of the job. Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream. If the standard input is set it specifies the network path of the job's input stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'i', the input file will be fetched by Altair Grid Engine from the specified host or from the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for bulk job submission, (See also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be read the job enters the state DRMAA_PS_FAILED.

drmaa_output_path - "[<hostname>]:<file_path>"

Specifies the standard output of the job. If not explicitly set in the job template, the whereabouts of the job's output stream is not defined. If set, this attribute specifies the network path of the job's output stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'o', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the

file_path denotes the remaining portion of *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.

drmaa_error_path - "[<hostname>]:<file_path>"

Specifies the standard error of the job. If not explicitly set in the job template, the whereabouts of the job's error stream is not defined. If set, this attribute specifies the network path of the job's error stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'e', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED. The attribute name is drmaa_error_path.

drmaa_join_files - "{y|n}"

Specifies if the job's error stream should be intermixed with the output stream. If not explicitly set in the job template the attribute defaults to 'n'. Either 'y' or 'n' can be specified. If 'y' is specified Altair Grid Engine will ignore the value of the 'drmaa_error_path' job template attribute and intermix the standard error stream with the standard output stream as specified with 'drmaa_output_path'.

drmaa_v_argv - "argv1 argv2 . . ."

Specifies the arguments to the job.

drmaa_job_category - "<category>"

Specifies the DRMAA job category. The *category* string is used by Altair Grid Engine as a reference into the *qtask* file. Certain *qsub*(1) options used in the referenced qtask file line are applied to the job template before submission to allow site-specific resolving of resources and/or policies. The cluster qtask file, the local qtask file, and the user qtask file

are searched. Job settings resulting from job template category are overridden by settings resulting from the job template drmaa_native_specification attribute as well as by explicit DRMAA job template settings.

In order to avoid collisions with command names in the qtask files, it is recommended that DRMAA job category names take the form: <category_name>.cat.

The options -help, -sync, -t, -verify, and -w w|v are ignored. The -cwd option is ignored unless the $SGE_DRMAA_ALLOW_CWD environment variable is set.

drmaa_native_specification - "<native_specification>"

Specifies Altair Grid Engine native *qsub*(1) options which will be interpreted as part of the DRMAA job template. All options available to *qsub*(1) command may be used in the *native_specification*, except for -help, -sync, -t, -verify, and -w w|v. The -cwd option may only be used if the SGE_DRMAA_ALLOW_CWD environment variable is set. This is because the current parsing algorithm for -cwd is not thread-safe. Options set in the *native specification* will be overridden by the corresponding DRMAA attributes. See *qsub*(1) for more information on qsub options.

drmaa_v_env - "<name1>=<value1> <name2>=<value2> ..."

Specifies the job environment. Each environment *value* defines the remote environment. The *value* overrides the remote environment values if there is a collision.

drmaa_v_email - "<email1> <email2> ..."

Specifies e-mail addresses that are used to report the job completion and status.

drmaa_block_email - "{0|1}"

Specifies whether e-mail sending shall blocked or not. By default email is not sent. If, however, a setting in a cluster or user settings file or the 'drmaa_native_specification' or 'drmaa_job_category' attribute enables sending email in association with job events, the 'drmaa_block_email' attribute will override that setting, causing no email to be sent.

drmaa_start_time - "[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]"

Specifies the earliest time when the job may be eligible to be run where

```
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
```

```
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
```

If the optional UTC-offset is not specified, the offset associated with the local timezone will be used. If the day (DD) is not specified, the current day will be used unless the specified hour:mm:ss has already elapsed, in which case the next day will be used. Similarly for month (MM), year (YY), and century-1 (CC). Example: The time: Sep 3 4:47:27 PM PDT 2002, could be represented as: 2002/09/03 16:47:27 -07:00.

drmaa_transfer_files - "[i][o][e]"

Specifies, which of the standard I/O files (stdin, stdout and stderr) are to be transferred to/from the execution host. If not set, defaults to ""‘. Any combination of ‘e’, ‘i’ and ‘o’ may be specified. See drmaa_input_path, drmaa_output_path and drmaa_error_path for information about how to specify the standard input file, standard output file and standard error file. The file transfer mechanism itself must be configured by the administrator (see *sge_conf*(5)). When it is configured, the administrator has to enable drmaa_transfer_files. If it is not configured,"drmaa_transfer_files" is not enabled and can't be used.

ENVIRONMENTAL VARIABLES

- *SGE_DRMAA_ALLOW_CWD* Enables the parsing of the -cwd option from the sge_request file(s), job category, and/or the native specification attribute. This option is disabled by default because the algorithm for parsing the -cwd option is not thread-safe.
- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_next_attr_name() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_jobtemplate(3) and *drmaa_submit*(3).

COPYRIGHT

drmaa_get_contact

NAME

drmaa_strerror, drmaa_get_contact, drmaa_version, drmaa_get_DRM_system - Miscellaneous DRMAA functions.

SYNOPSIS

```
#include "drmaa.h"

const char *drmaa_strerror(
        int drmaa_errno
);

int drmaa_get_contact(
        char *contact,
        size_t contact_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_version(
        unsigned int *major,
        unsigned int *minor,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRM_system(
        char *drm_system,
        size_t drm_system_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRMAA_implementation(
        char *drm_impl,
        size_t drm_impl_len,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_strerror() function returns a message text associated with the DRMAA error number,*drmaa_errno*. For invalid DRMAA error codes 'NULL' is returned.

drmaa_get_contact()

The drmaa_get_contact() returns an opaque string containing contact information related to the current DRMAA session to be used with the *drmaa_init*(3) function. The opaque string contains the information required by drmaa_init() to reconnect to the current session instead of creating a new session. *drmaa_init*(3) function.

The drmaa_get_contact() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_version()

The drmaa_version() function returns into the integers pointed to by *major* and *minor*, the major and minor version numbers of the DRMAA library. For a DRMAA 1.0 compliant implementation '1' and '0' will be returned in *major* and *minor*, respectively.

drmaa_get_DRM_system()

The drmaa_get_DRM_system() function returns into *drm_system* up to *drm_system_len* characters of a string containing Altair Grid Engine product and version information.

The drmaa_get_DRM_system() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_get_DRMAA_implementation()

The drmaa_get_DRMAA_implementation() function returns into *drm_system* up to *drm_system_len* characters of a string containing the Altair Grid Engine DRMAA implementation version information. In the current implementation, the drmaa_get_DRMAA_implementation() function returns the same result as the drmaa_get_DRM_system() function.

The drmaa_get_DRMAA_implementation() function returns the same value before and after *drmaa_init*(3) is called.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_contact(), drmaa_version(), and drmaa_get_DRM_system() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_contact(), drmaa_version(), drmaa_get_DRM_system(), and drmaa_get_DRMAA_implementation will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_contact() and drmaa_get_DRM_system() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

SEE ALSO

drmaa_session(3).

COPYRIGHT

drmaa_get_next_attr_name

NAME

drmaa_get_attribute_names, drmaa_get_vector_attribute_names, drmaa_get_next_attr_name, drmaa_get_num_attr_names, drmaa_release_attr_names - DRMAA job template attributes

SYNOPSIS

```
#include "drmaa.h"

int drmaa_get_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_name(
        drmaa_attr_names_t* values,
        char *value,
        int value_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_names(
        drmaa_attr_names_t* values,
        int *size
);

void drmaa_release_attr_names(
        drmaa_attr_names_t* values
);
```

DESCRIPTION

The drmaa_get_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported non-vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The number of names in the names string vector can be determined using *drmaa_get_num_attr_names*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_attribute*(3) and *drmaa_get_attribute*(3) for setting and inspecting non-vector attributes.

drmaa_get_vector_attribute_names()

The drmaa_get_vector_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_vector_attribute*(3) and *drmaa_get_vector_attribute*(3) for setting and inspecting vector attributes.

drmaa_get_next_attr_name()

Each time drmaa_get_next_attr_name() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA names string vector, *values*. Once the names list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_names()

The drmaa_get_num_attr_names() returns into *size* the number of entries in the DRMAA names string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_names()

The drmaa_release_attr_names() function releases all resources associated with the DRMAA names string vector, *values*.

Attribute Priorities

DRMAA job template attributes can be set from six different sources. In order of precedence, from lowest to highest, these are: options set by DRMAA automatically by default, options set in the *sge_request*(5) file(s), options set in the script file, options set by the drmaa_job_category attribute, options set by the drmaa_native_specification attribute, and options set through other DRMAA attributes.

By default DRMAA sets four options for all jobs. These are “-p 0”, “-b yes”, “-shell no”, and “-w e”. This means that by default, all jobs will have priority 0, all jobs will be treated as binary, i.e. no scripts args will be parsed, all jobs will be executed without a wrapper shell, and jobs which are unschedulable will cause a submit error.

The *sge_request*(5) file, found in the $SGE_ROOT/$SGE_CELL/common directory, may contain options to be applied to all jobs. The .sge_request file found in the user’s home directory or the current working directory may also contain options to be applied to certain jobs. See *sge_request*(5) for more information. If the *sge_request*(5) file contains “-b no” or if the drmaa_native_specification attribute is set and contains “-b no”, the script file will be parsed for in-line arguments. Otherwise, no scripts args will be interpreted. See *qsub*(1) for more information. If the drmaa_job_category attribute is set, and the category it points to exists in one of the *qtask* files, the options associated with that category will be applied to the job template.

If the drmaa_native_specification attribute is set, all options contained therein will be applied to the job template. See the drmaa_native_specification below for more information. Other DRMAA attributes will override any previous settings. For example, if the sge_request file contains “-j y”, but the drmaa_join_files attribute is set to “n”, the ultimate result is that the input and output files will remain separate.

For various reasons, some options are silently ignored by DRMAA. Setting any of these options will have no effect. The ignored options are: -cwd, -help, -sync, -t, -verify, -w w, and -w v. The -cwd option can be re-enabled by setting the environment variable, SGE_DRMAA_ALLOW_CWD. However, the -cwd option is not thread safe and should not be used in a multi-threaded context.

Attribute Correlations

The following DRMAA attributes correspond to the following *qsub*(1) options:

```
DRMAA Attribute                  qsub Option
-----------------------------------------------------
drmaa_remote_command             script file
drmaa_v_argv                     script file args
drmaa_js_state = "drmaa_hold"    -h
drmaa_v_env                      -v
drmaa_wd = \$PWD                  -cwd
drmaa_job_category               (qtsch qtask)*
drmaa_native_specification       ALL*
drmaa_v_email                    -M
drmaa_block_email = "1"          -m n
drmaa_start_time                 -a
drmaa_job_name                   -N
drmaa_input_path                 -i
drmaa_output_path                -o
drmaa_error_path                 -e
drmaa_join_files                 -j
drmaa_transfer_files             (prolog and epilog)*
```

```
* See the individual attribute description below
```

DRMAA JOB TEMPLATE ATTRIBUTES

drmaa_remote_command - "<remote_command>"

Specifies the remote command to execute. The *remote_command* must be the path of an executable that is available at the job's execution host. If the path is relative, it is assumed to be relative to the working directory, usually set through the drmaa_wd attribute. If working directory is not set, the path is assumed to be relative to the user's home directory.

The file pointed to by remote_command may either be an executable binary or an executable script. If a script, it must include the path to the shell in a #! line at the beginning of the script. By default, the remote command will be executed directly, as by *exec*(2). To have the remote command executed in a shell, such as to preserve environment settings, use the drmaa_native_specification attribute to include the "-shell yes" option. Jobs which are executed by a wrapper shell fail differently from jobs which are executed directly. When a job which contains a user error, such as an invalid path to the executable, is executed by a wrapper shell, the job will execute successfully, but exit with a return code of 1. When a job which contains such an error is executed directly, it will enter the DRMAA_PS_FAILED state upon execution.

drmaa_js_state - "{drmaa_hold|drmaa_active}"

Specifies the job state at submission. The string values 'drmaa_hold' and 'drmaa_active' are supported. When 'drmaa_active' is used the job is submitted in a runnable state. When 'drmaa_hold' is used the job is submitted in user hold state (either DRMAA_PS_USER_ON_HOLD or DRMAA_PS_USER_SYSTEM_ON_HOLD). This attribute is largely equivalent to the *qsub*(1) submit option '-h'.

drmaa_wd - "<directory_name>"

Specifies the directory name where the job will be executed. A '$drmaa_hd_ph$' placeholder at the beginning of the *directory_name* denotes the remaining string portion as a relative directory name that is resolved relative to the job user's home directory at the execution host. When the DRMAA job template is used for bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *directory_name* to cause a substitution with the parametric job's index. The *directory_name* must be specified in a syntax that is common at the host where the job is executed. If set to a relative path and no placeholder is used, a path relative to the user's home directory is assumed. If not set, the working directory will default to the user's home directory. If set and the given directory does not exist the job will enter the DRMAA_PS_FAILED state when run.

Note that the working directory path is the path on the execution host. If binary mode is disabled, an attempt to find the job script will be made, relative to the working directory

path. That means that the path to the script must be the same on both the submission and execution hosts.

drmaa_job_name - "<job_name>"

Specifies the job's name. Setting the job name is equivalent to use of *qsub*(1) submit option '-N' with *job_name* as option argument.

drmaa_input_path - "[<hostname>]:<file_path>"

Specifies the standard input of the job. Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream. If the standard input is set it specifies the network path of the job's input stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'i', the input file will be fetched by Altair Grid Engine from the specified host or from the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for bulk job submission, (See also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be read the job enters the state DRMAA_PS_FAILED.

drmaa_output_path - "[<hostname>]:<file_path>"

Specifies the standard output of the job. If not explicitly set in the job template, the whereabouts of the job's output stream is not defined. If set, this attribute specifies the network path of the job's output stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'o', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the

file_path denotes the remaining portion of *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.

drmaa_error_path - "[<hostname>]:<file_path>"

Specifies the standard error of the job. If not explicitly set in the job template, the whereabouts of the job's error stream is not defined. If set, this attribute specifies the network path of the job's error stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'e', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED. The attribute name is drmaa_error_path.

drmaa_join_files - "{y|n}"

Specifies if the job's error stream should be intermixed with the output stream. If not explicitly set in the job template the attribute defaults to 'n'. Either 'y' or 'n' can be specified. If 'y' is specified Altair Grid Engine will ignore the value of the 'drmaa_error_path' job template attribute and intermix the standard error stream with the standard output stream as specified with 'drmaa_output_path'.

drmaa_v_argv - "argv1 argv2 . . ."

Specifies the arguments to the job.

drmaa_job_category - "<category>"

Specifies the DRMAA job category. The *category* string is used by Altair Grid Engine as a reference into the *qtask* file. Certain *qsub*(1) options used in the referenced qtask file line are applied to the job template before submission to allow site-specific resolving of resources and/or policies. The cluster qtask file, the local qtask file, and the user qtask file

are searched. Job settings resulting from job template category are overridden by settings resulting from the job template drmaa_native_specification attribute as well as by explicit DRMAA job template settings.

In order to avoid collisions with command names in the qtask files, it is recommended that DRMAA job category names take the form: <category_name>.cat.

The options -help, -sync, -t, -verify, and -w w|v are ignored. The -cwd option is ignored unless the $SGE_DRMAA_ALLOW_CWD environment variable is set.

drmaa_native_specification - "<native_specification>"

Specifies Altair Grid Engine native *qsub*(1) options which will be interpreted as part of the DRMAA job template. All options available to *qsub*(1) command may be used in the *native_specification*, except for -help, -sync, -t, -verify, and -w w|v. The -cwd option may only be used if the SGE_DRMAA_ALLOW_CWD environment variable is set. This is because the current parsing algorithm for -cwd is not thread-safe. Options set in the *native specification* will be overridden by the corresponding DRMAA attributes. See *qsub*(1) for more information on qsub options.

drmaa_v_env - "<name1>=<value1> <name2>=<value2> ..."

Specifies the job environment. Each environment *value* defines the remote environment. The *value* overrides the remote environment values if there is a collision.

drmaa_v_email - "<email1> <email2> ..."

Specifies e-mail addresses that are used to report the job completion and status.

drmaa_block_email - "{0|1}"

Specifies whether e-mail sending shall blocked or not. By default email is not sent. If, however, a setting in a cluster or user settings file or the 'drmaa_native_specification' or 'drmaa_job_category' attribute enables sending email in association with job events, the 'drmaa_block_email' attribute will override that setting, causing no email to be sent.

drmaa_start_time - "[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]"

Specifies the earliest time when the job may be eligible to be run where

```
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
```

```
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
```

If the optional UTC-offset is not specified, the offset associated with the local timezone will be used. If the day (DD) is not specified, the current day will be used unless the specified hour:mm:ss has already elapsed, in which case the next day will be used. Similarly for month (MM), year (YY), and century-1 (CC). Example: The time: Sep 3 4:47:27 PM PDT 2002, could be represented as: 2002/09/03 16:47:27 -07:00.

drmaa_transfer_files - "[i][o][e]"

Specifies, which of the standard I/O files (stdin, stdout and stderr) are to be transferred to/from the execution host. If not set, defaults to "". Any combination of 'e', 'i' and 'o' may be specified. See drmaa_input_path, drmaa_output_path and drmaa_error_path for information about how to specify the standard input file, standard output file and standard error file. The file transfer mechanism itself must be configured by the administrator (see *sge_conf*(5)). When it is configured, the administrator has to enable drmaa_transfer_files. If it is not configured,"drmaa_transfer_files" is not enabled and can't be used.

ENVIRONMENTAL VARIABLES

- *SGE_DRMAA_ALLOW_CWD* Enables the parsing of the -cwd option from the sge_request file(s), job category, and/or the native specification attribute. This option is disabled by default because the algorithm for parsing the -cwd option is not thread-safe.
- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_next_attr_name() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_jobtemplate(3) and *drmaa_submit*(3).

COPYRIGHT

drmaa_get_next_attr_value

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_get_next_job_id

NAME

drmaa_run_job, drmaa_run_bulk_jobs, drmaa_get_next_job_id, drmaa_get_num_job_ids, drmaa_release_job_ids - Job submission

SYNOPSIS

```
#include "drmaa.h"

int drmaa_run_job(
        char *job_id,
        size_t job_id_len,
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_run_bulk_jobs(
        drmaa_job_ids_t **jobids,
        drmaa_job_template_t *jt,
        int start,
        int end,
        int incr,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_job_id(
        drmaa_job_ids_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_job_ids(
        drmaa_job_ids_t* values,
        int *size
);

void drmaa_release_job_ids(
        drmaa_job_ids_t* values
);
```

DESCRIPTION

drmaa_run_job() submits an xAltair Grid Engine job with attributes defined in the DRMAA job template, *jt*. On success up to *job_id_len* bytes of the job identifier are returned into the buffer, *job_id*.

drmaa_run_bulk_jobs()

The drmaa_run_bulk_jobs() submits a Altair Grid Engine array job very much as if the *qsub*(1) option '-t *start*-*end*:*incr*' had been used along with the additional attributes defined in the DRMAA job template, *jt*. The same constraints regarding value ranges are also in effect for the parameters *start*, *end*, and *incr* as for *qsub*(1) -t. On success a DRMAA job id string vector containing job identifiers for each array job task is returned into *jobids*. The job identifiers in the job id string vector can be extracted using *drmaa_get_next_job_id*(3). The number of identifiers in the job id string vector can be determined using *drmaa_get_num_job_ids*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the job id string vector returned into *jobids* using *drmaa_release_job_ids*(3).

drmaa_get_next_job_id()

Each time drmaa_get_next_job_id() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA job id string vector, *values*. Once the job ids list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_job_ids()

The drmaa_get_num_job_ids() returns into *size* the number of entries in the DRMAA job ids string vector. This function is only available in the 1.0 implementation.

drmaa_release_job_ids()

The drmaa_release_attr_job_id() function releases all resources associated with the DRMAA job id string vector, *values*. This operation has no effect on the actual xAltair Grid Engine array job tasks.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default xAltair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_run_job() and drmaa_run_bulk_jobs() functions will fail if:

DRMAA_ERRNO_TRY_LATER

The DRM system indicated that it is too busy to accept the job. A retry may succeed, however.

DRMAA_ERRNO_DENIED_BY_DRM

The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

The drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

ENVIRONMENTAL VARIABLES

SGE_DRMAA_ENABLE_ERROR_STATE

When this environment variable is set, then jobs that are submitted with drmaa_run_job() or drmaa_run_bulk_jobs() will change into error state when either during the job start or during the execution of the job an error occurs. Normally DRMAA jobs will not switch into error state when something fails.

SEE ALSO

drmaa_attributes(3), *drmaa_jobtemplate*(3).

COPYRIGHT

drmaa_get_num_attr_names

NAME

drmaa_get_attribute_names, drmaa_get_vector_attribute_names, drmaa_get_next_attr_name, drmaa_get_num_attr_names, drmaa_release_attr_names - DRMAA job template attributes

SYNOPSIS

```
#include "drmaa.h"

int drmaa_get_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_name(
        drmaa_attr_names_t* values,
        char *value,
        int value_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_names(
        drmaa_attr_names_t* values,
        int *size
);

void drmaa_release_attr_names(
        drmaa_attr_names_t* values
);
```

DESCRIPTION

The drmaa_get_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported non-vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The number of names in the names string vector can be determined using *drmaa_get_num_attr_names*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_attribute*(3) and *drmaa_get_attribute*(3) for setting and inspecting non-vector attributes.

drmaa_get_vector_attribute_names()

The drmaa_get_vector_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_vector_attribute*(3) and *drmaa_get_vector_attribute*(3) for setting and inspecting vector attributes.

drmaa_get_next_attr_name()

Each time drmaa_get_next_attr_name() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA names string vector, *values*. Once the names list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_names()

The drmaa_get_num_attr_names() returns into *size* the number of entries in the DRMAA names string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_names()

The drmaa_release_attr_names() function releases all resources associated with the DRMAA names string vector, *values*.

Attribute Priorities

DRMAA job template attributes can be set from six different sources. In order of precedence, from lowest to highest, these are: options set by DRMAA automatically by default, options set in the *sge_request*(5) file(s), options set in the script file, options set by the drmaa_job_category attribute, options set by the drmaa_native_specification attribute, and options set through other DRMAA attributes.

By default DRMAA sets four options for all jobs. These are “-p 0”, “-b yes”, “-shell no”, and “-w e”. This means that by default, all jobs will have priority 0, all jobs will be treated as binary, i.e. no scripts args will be parsed, all jobs will be executed without a wrapper shell, and jobs which are unschedulable will cause a submit error.

The *sge_request*(5) file, found in the $SGE_ROOT/$SGE_CELL/common directory, may contain options to be applied to all jobs. The .sge_request file found in the user’s home directory or the current working directory may also contain options to be applied to certain jobs. See *sge_request*(5) for more information. If the *sge_request*(5) file contains “-b no” or if the drmaa_native_specification attribute is set and contains “-b no”, the script file will be parsed for in-line arguments. Otherwise, no scripts args will be interpreted. See *qsub*(1) for more information. If the drmaa_job_category attribute is set, and the category it points to exists in one of the *qtask* files, the options associated with that category will be applied to the job template.

If the drmaa_native_specification attribute is set, all options contained therein will be applied to the job template. See the drmaa_native_specification below for more information. Other DRMAA attributes will override any previous settings. For example, if the sge_request file contains “-j y”, but the drmaa_join_files attribute is set to “n”, the ultimate result is that the input and output files will remain separate.

For various reasons, some options are silently ignored by DRMAA. Setting any of these options will have no effect. The ignored options are: -cwd, -help, -sync, -t, -verify, -w w, and -w v. The -cwd option can be re-enabled by setting the environment variable, SGE_DRMAA_ALLOW_CWD. However, the -cwd option is not thread safe and should not be used in a multi-threaded context.

Attribute Correlations

The following DRMAA attributes correspond to the following *qsub*(1) options:

```
DRMAA Attribute                  qsub Option
-----------------------------------------------------
drmaa_remote_command             script file
drmaa_v_argv                     script file args
drmaa_js_state = "drmaa_hold"    -h
drmaa_v_env                      -v
drmaa_wd = \$PWD                  -cwd
drmaa_job_category               (qtsch qtask)*
drmaa_native_specification       ALL*
drmaa_v_email                    -M
drmaa_block_email = "1"          -m n
drmaa_start_time                 -a
drmaa_job_name                   -N
drmaa_input_path                 -i
drmaa_output_path                -o
drmaa_error_path                 -e
drmaa_join_files                 -j
drmaa_transfer_files             (prolog and epilog)*
```

```
* See the individual attribute description below
```

DRMAA JOB TEMPLATE ATTRIBUTES

drmaa_remote_command - "<remote_command>"

Specifies the remote command to execute. The *remote_command* must be the path of an executable that is available at the job's execution host. If the path is relative, it is assumed to be relative to the working directory, usually set through the drmaa_wd attribute. If working directory is not set, the path is assumed to be relative to the user's home directory.

The file pointed to by remote_command may either be an executable binary or an executable script. If a script, it must include the path to the shell in a #! line at the beginning of the script. By default, the remote command will be executed directly, as by *exec*(2). To have the remote command executed in a shell, such as to preserve environment settings, use the drmaa_native_specification attribute to include the "-shell yes" option. Jobs which are executed by a wrapper shell fail differently from jobs which are executed directly. When a job which contains a user error, such as an invalid path to the executable, is executed by a wrapper shell, the job will execute successfully, but exit with a return code of 1. When a job which contains such an error is executed directly, it will enter the DRMAA_PS_FAILED state upon execution.

drmaa_js_state - "{drmaa_hold|drmaa_active}"

Specifies the job state at submission. The string values 'drmaa_hold' and 'drmaa_active' are supported. When 'drmaa_active' is used the job is submitted in a runnable state. When 'drmaa_hold' is used the job is submitted in user hold state (either DRMAA_PS_USER_ON_HOLD or DRMAA_PS_USER_SYSTEM_ON_HOLD). This attribute is largely equivalent to the *qsub*(1) submit option '-h'.

drmaa_wd - "<directory_name>"

Specifies the directory name where the job will be executed. A '$drmaa_hd_ph$' placeholder at the beginning of the *directory_name* denotes the remaining string portion as a relative directory name that is resolved relative to the job user's home directory at the execution host. When the DRMAA job template is used for bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *directory_name* to cause a substitution with the parametric job's index. The *directory_name* must be specified in a syntax that is common at the host where the job is executed. If set to a relative path and no placeholder is used, a path relative to the user's home directory is assumed. If not set, the working directory will default to the user's home directory. If set and the given directory does not exist the job will enter the DRMAA_PS_FAILED state when run.

Note that the working directory path is the path on the execution host. If binary mode is disabled, an attempt to find the job script will be made, relative to the working directory

path. That means that the path to the script must be the same on both the submission and execution hosts.

drmaa_job_name - "<job_name>"

Specifies the job's name. Setting the job name is equivalent to use of *qsub*(1) submit option '-N' with *job_name* as option argument.

drmaa_input_path - "[<hostname>]:<file_path>"

Specifies the standard input of the job. Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream. If the standard input is set it specifies the network path of the job's input stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'i', the input file will be fetched by Altair Grid Engine from the specified host or from the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for bulk job submission, (See also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be read the job enters the state DRMAA_PS_FAILED.

drmaa_output_path - "[<hostname>]:<file_path>"

Specifies the standard output of the job. If not explicitly set in the job template, the whereabouts of the job's output stream is not defined. If set, this attribute specifies the network path of the job's output stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'o', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the

file_path denotes the remaining portion of *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.

drmaa_error_path - "[<hostname>]:<file_path>"

Specifies the standard error of the job. If not explicitly set in the job template, the whereabouts of the job's error stream is not defined. If set, this attribute specifies the network path of the job's error stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'e', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED. The attribute name is drmaa_error_path.

drmaa_join_files - "{y|n}"

Specifies if the job's error stream should be intermixed with the output stream. If not explicitly set in the job template the attribute defaults to 'n'. Either 'y' or 'n' can be specified. If 'y' is specified Altair Grid Engine will ignore the value of the 'drmaa_error_path' job template attribute and intermix the standard error stream with the standard output stream as specified with 'drmaa_output_path'.

drmaa_v_argv - "argv1 argv2 . . ."

Specifies the arguments to the job.

drmaa_job_category - "<category>"

Specifies the DRMAA job category. The *category* string is used by Altair Grid Engine as a reference into the *qtask* file. Certain *qsub*(1) options used in the referenced qtask file line are applied to the job template before submission to allow site-specific resolving of resources and/or policies. The cluster qtask file, the local qtask file, and the user qtask file

are searched. Job settings resulting from job template category are overridden by settings resulting from the job template drmaa_native_specification attribute as well as by explicit DRMAA job template settings.

In order to avoid collisions with command names in the qtask files, it is recommended that DRMAA job category names take the form: <category_name>.cat.

The options -help, -sync, -t, -verify, and -w w|v are ignored. The -cwd option is ignored unless the $SGE_DRMAA_ALLOW_CWD environment variable is set.

drmaa_native_specification - "<native_specification>"

Specifies Altair Grid Engine native *qsub*(1) options which will be interpreted as part of the DRMAA job template. All options available to *qsub*(1) command may be used in the *native_specification*, except for -help, -sync, -t, -verify, and -w w|v. The -cwd option may only be used if the SGE_DRMAA_ALLOW_CWD environment variable is set. This is because the current parsing algorithm for -cwd is not thread-safe. Options set in the *native specification* will be overridden by the corresponding DRMAA attributes. See *qsub*(1) for more information on qsub options.

drmaa_v_env - "<name1>=<value1> <name2>=<value2> ..."

Specifies the job environment. Each environment *value* defines the remote environment. The *value* overrides the remote environment values if there is a collision.

drmaa_v_email - "<email1> <email2> ..."

Specifies e-mail addresses that are used to report the job completion and status.

drmaa_block_email - "{0|1}"

Specifies whether e-mail sending shall blocked or not. By default email is not sent. If, however, a setting in a cluster or user settings file or the 'drmaa_native_specification' or 'drmaa_job_category' attribute enables sending email in association with job events, the 'drmaa_block_email' attribute will override that setting, causing no email to be sent.

drmaa_start_time - "[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]"

Specifies the earliest time when the job may be eligible to be run where

```
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
```

```
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
```

If the optional UTC-offset is not specified, the offset associated with the local timezone will be used. If the day (DD) is not specified, the current day will be used unless the specified hour:mm:ss has already elapsed, in which case the next day will be used. Similarly for month (MM), year (YY), and century-1 (CC). Example: The time: Sep 3 4:47:27 PM PDT 2002, could be represented as: 2002/09/03 16:47:27 -07:00.

drmaa_transfer_files - "[i][o][e]"

Specifies, which of the standard I/O files (stdin, stdout and stderr) are to be transferred to/from the execution host. If not set, defaults to "". Any combination of 'e', 'i' and 'o' may be specified. See drmaa_input_path, drmaa_output_path and drmaa_error_path for information about how to specify the standard input file, standard output file and standard error file. The file transfer mechanism itself must be configured by the administrator (see *sge_conf*(5)). When it is configured, the administrator has to enable drmaa_transfer_files. If it is not configured,"drmaa_transfer_files" is not enabled and can't be used.

ENVIRONMENTAL VARIABLES

- *SGE_DRMAA_ALLOW_CWD* Enables the parsing of the -cwd option from the sge_request file(s), job category, and/or the native specification attribute. This option is disabled by default because the algorithm for parsing the -cwd option is not thread-safe.
- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_next_attr_name() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_jobtemplate(3) and *drmaa_submit*(3).

COPYRIGHT

drmaa_get_num_attr_values

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_get_num_job_ids

NAME

drmaa_run_job, drmaa_run_bulk_jobs, drmaa_get_next_job_id, drmaa_get_num_job_ids, drmaa_release_job_ids - Job submission

SYNOPSIS

```
#include "drmaa.h"

int drmaa_run_job(
        char *job_id,
        size_t job_id_len,
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_run_bulk_jobs(
        drmaa_job_ids_t **jobids,
        drmaa_job_template_t *jt,
        int start,
        int end,
        int incr,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_job_id(
        drmaa_job_ids_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_job_ids(
        drmaa_job_ids_t* values,
        int *size
);

void drmaa_release_job_ids(
        drmaa_job_ids_t* values
);
```

DESCRIPTION

drmaa_run_job() submits an xAltair Grid Engine job with attributes defined in the DRMAA job template, *jt*. On success up to *job_id_len* bytes of the job identifier are returned into the buffer, *job_id*.

drmaa_run_bulk_jobs()

The drmaa_run_bulk_jobs() submits a Altair Grid Engine array job very much as if the *qsub*(1) option '-t *start*-*end*:*incr*' had been used along with the additional attributes defined in the DRMAA job template, *jt*. The same constraints regarding value ranges are also in effect for the parameters *start*, *end*, and *incr* as for *qsub*(1) -t. On success a DRMAA job id string vector containing job identifiers for each array job task is returned into *jobids*. The job identifiers in the job id string vector can be extracted using *drmaa_get_next_job_id*(3). The number of identifiers in the job id string vector can be determined using *drmaa_get_num_job_ids*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the job id string vector returned into *jobids* using *drmaa_release_job_ids*(3).

drmaa_get_next_job_id()

Each time drmaa_get_next_job_id() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA job id string vector, *values*. Once the job ids list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_job_ids()

The drmaa_get_num_job_ids() returns into *size* the number of entries in the DRMAA job ids string vector. This function is only available in the 1.0 implementation.

drmaa_release_job_ids()

The drmaa_release_attr_job_id() function releases all resources associated with the DRMAA job id string vector, *values*. This operation has no effect on the actual xAltair Grid Engine array job tasks.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default xAltair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_run_job() and drmaa_run_bulk_jobs() functions will fail if:

DRMAA_ERRNO_TRY_LATER

The DRM system indicated that it is too busy to accept the job. A retry may succeed, however.

DRMAA_ERRNO_DENIED_BY_DRM

The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

The drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

ENVIRONMENTAL VARIABLES

SGE_DRMAA_ENABLE_ERROR_STATE

When this environment variable is set, then jobs that are submitted with drmaa_run_job() or drmaa_run_bulk_jobs() will change into error state when either during the job start or during the execution of the job an error occurs. Normally DRMAA jobs will not switch into error state when something fails.

SEE ALSO

drmaa_attributes(3), *drmaa_jobtemplate*(3).

COPYRIGHT

drmaa_get_vector_attribute

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_get_vector_attribute_names

NAME

drmaa_get_attribute_names, drmaa_get_vector_attribute_names, drmaa_get_next_attr_name, drmaa_get_num_attr_names, drmaa_release_attr_names - DRMAA job template attributes

SYNOPSIS

```
#include "drmaa.h"

int drmaa_get_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_name(
        drmaa_attr_names_t* values,
        char *value,
        int value_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_names(
        drmaa_attr_names_t* values,
        int *size
);

void drmaa_release_attr_names(
        drmaa_attr_names_t* values
);
```

DESCRIPTION

The drmaa_get_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported non-vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The number of names in the names string vector can be determined using *drmaa_get_num_attr_names*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_attribute*(3) and *drmaa_get_attribute*(3) for setting and inspecting non-vector attributes.

drmaa_get_vector_attribute_names()

The drmaa_get_vector_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_vector_attribute*(3) and *drmaa_get_vector_attribute*(3) for setting and inspecting vector attributes.

drmaa_get_next_attr_name()

Each time drmaa_get_next_attr_name() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA names string vector, *values*. Once the names list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_names()

The drmaa_get_num_attr_names() returns into *size* the number of entries in the DRMAA names string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_names()

The drmaa_release_attr_names() function releases all resources associated with the DRMAA names string vector, *values*.

Attribute Priorities

DRMAA job template attributes can be set from six different sources. In order of precedence, from lowest to highest, these are: options set by DRMAA automatically by default, options set in the *sge_request*(5) file(s), options set in the script file, options set by the drmaa_job_category attribute, options set by the drmaa_native_specification attribute, and options set through other DRMAA attributes.

By default DRMAA sets four options for all jobs. These are “-p 0”, “-b yes”, “-shell no”, and “-w e”. This means that by default, all jobs will have priority 0, all jobs will be treated as binary, i.e. no scripts args will be parsed, all jobs will be executed without a wrapper shell, and jobs which are unschedulable will cause a submit error.

The *sge_request*(5) file, found in the $SGE_ROOT/$SGE_CELL/common directory, may contain options to be applied to all jobs. The .sge_request file found in the user’s home directory or the current working directory may also contain options to be applied to certain jobs. See *sge_request*(5) for more information. If the *sge_request*(5) file contains “-b no” or if the drmaa_native_specification attribute is set and contains “-b no”, the script file will be parsed for in-line arguments. Otherwise, no scripts args will be interpreted. See *qsub*(1) for more information. If the drmaa_job_category attribute is set, and the category it points to exists in one of the *qtask* files, the options associated with that category will be applied to the job template.

If the drmaa_native_specification attribute is set, all options contained therein will be applied to the job template. See the drmaa_native_specification below for more information. Other DRMAA attributes will override any previous settings. For example, if the sge_request file contains “-j y”, but the drmaa_join_files attribute is set to “n”, the ultimate result is that the input and output files will remain separate.

For various reasons, some options are silently ignored by DRMAA. Setting any of these options will have no effect. The ignored options are: -cwd, -help, -sync, -t, -verify, -w w, and -w v. The -cwd option can be re-enabled by setting the environment variable, SGE_DRMAA_ALLOW_CWD. However, the -cwd option is not thread safe and should not be used in a multi-threaded context.

Attribute Correlations

The following DRMAA attributes correspond to the following *qsub*(1) options:

```
DRMAA Attribute                  qsub Option
-----------------------------------------------------
drmaa_remote_command             script file
drmaa_v_argv                     script file args
drmaa_js_state = "drmaa_hold"    -h
drmaa_v_env                      -v
drmaa_wd = \$PWD                  -cwd
drmaa_job_category               (qtsch qtask)*
drmaa_native_specification       ALL*
drmaa_v_email                    -M
drmaa_block_email = "1"          -m n
drmaa_start_time                 -a
drmaa_job_name                   -N
drmaa_input_path                 -i
drmaa_output_path                -o
drmaa_error_path                 -e
drmaa_join_files                 -j
drmaa_transfer_files             (prolog and epilog)*
```

```
* See the individual attribute description below
```

DRMAA JOB TEMPLATE ATTRIBUTES

drmaa_remote_command - "<remote_command>"

Specifies the remote command to execute. The *remote_command* must be the path of an executable that is available at the job's execution host. If the path is relative, it is assumed to be relative to the working directory, usually set through the drmaa_wd attribute. If working directory is not set, the path is assumed to be relative to the user's home directory.

The file pointed to by remote_command may either be an executable binary or an executable script. If a script, it must include the path to the shell in a #! line at the beginning of the script. By default, the remote command will be executed directly, as by *exec*(2). To have the remote command executed in a shell, such as to preserve environment settings, use the drmaa_native_specification attribute to include the "-shell yes" option. Jobs which are executed by a wrapper shell fail differently from jobs which are executed directly. When a job which contains a user error, such as an invalid path to the executable, is executed by a wrapper shell, the job will execute successfully, but exit with a return code of 1. When a job which contains such an error is executed directly, it will enter the DRMAA_PS_FAILED state upon execution.

drmaa_js_state - "{drmaa_hold|drmaa_active}"

Specifies the job state at submission. The string values 'drmaa_hold' and 'drmaa_active' are supported. When 'drmaa_active' is used the job is submitted in a runnable state. When 'drmaa_hold' is used the job is submitted in user hold state (either DRMAA_PS_USER_ON_HOLD or DRMAA_PS_USER_SYSTEM_ON_HOLD). This attribute is largely equivalent to the *qsub*(1) submit option '-h'.

drmaa_wd - "<directory_name>"

Specifies the directory name where the job will be executed. A '$drmaa_hd_ph$' placeholder at the beginning of the *directory_name* denotes the remaining string portion as a relative directory name that is resolved relative to the job user's home directory at the execution host. When the DRMAA job template is used for bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *directory_name* to cause a substitution with the parametric job's index. The *directory_name* must be specified in a syntax that is common at the host where the job is executed. If set to a relative path and no placeholder is used, a path relative to the user's home directory is assumed. If not set, the working directory will default to the user's home directory. If set and the given directory does not exist the job will enter the DRMAA_PS_FAILED state when run.

Note that the working directory path is the path on the execution host. If binary mode is disabled, an attempt to find the job script will be made, relative to the working directory

path. That means that the path to the script must be the same on both the submission and execution hosts.

drmaa_job_name - "<job_name>"

Specifies the job's name. Setting the job name is equivalent to use of *qsub*(1) submit option '-N' with *job_name* as option argument.

drmaa_input_path - "[<hostname>]:<file_path>"

Specifies the standard input of the job. Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream. If the standard input is set it specifies the network path of the job's input stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'i', the input file will be fetched by Altair Grid Engine from the specified host or from the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for bulk job submission, (See also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be read the job enters the state DRMAA_PS_FAILED.

drmaa_output_path - "[<hostname>]:<file_path>"

Specifies the standard output of the job. If not explicitly set in the job template, the whereabouts of the job's output stream is not defined. If set, this attribute specifies the network path of the job's output stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'o', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the

file_path denotes the remaining portion of *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.

drmaa_error_path - "[<hostname>]:<file_path>"

Specifies the standard error of the job. If not explicitly set in the job template, the whereabouts of the job's error stream is not defined. If set, this attribute specifies the network path of the job's error stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'e', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED. The attribute name is drmaa_error_path.

drmaa_join_files - "{y|n}"

Specifies if the job's error stream should be intermixed with the output stream. If not explicitly set in the job template the attribute defaults to 'n'. Either 'y' or 'n' can be specified. If 'y' is specified Altair Grid Engine will ignore the value of the 'drmaa_error_path' job template attribute and intermix the standard error stream with the standard output stream as specified with 'drmaa_output_path'.

drmaa_v_argv - "argv1 argv2 . . ."

Specifies the arguments to the job.

drmaa_job_category - "<category>"

Specifies the DRMAA job category. The *category* string is used by Altair Grid Engine as a reference into the *qtask* file. Certain *qsub*(1) options used in the referenced qtask file line are applied to the job template before submission to allow site-specific resolving of resources and/or policies. The cluster qtask file, the local qtask file, and the user qtask file

are searched. Job settings resulting from job template category are overridden by settings resulting from the job template drmaa_native_specification attribute as well as by explicit DRMAA job template settings.

In order to avoid collisions with command names in the qtask files, it is recommended that DRMAA job category names take the form: <category_name>.cat.

The options -help, -sync, -t, -verify, and -w w|v are ignored. The -cwd option is ignored unless the $SGE_DRMAA_ALLOW_CWD environment variable is set.

drmaa_native_specification - "<native_specification>"

Specifies Altair Grid Engine native *qsub*(1) options which will be interpreted as part of the DRMAA job template. All options available to *qsub*(1) command may be used in the *native_specification*, except for -help, -sync, -t, -verify, and -w w|v. The -cwd option may only be used if the SGE_DRMAA_ALLOW_CWD environment variable is set. This is because the current parsing algorithm for -cwd is not thread-safe. Options set in the *native specification* will be overridden by the corresponding DRMAA attributes. See *qsub*(1) for more information on qsub options.

drmaa_v_env - "<name1>=<value1> <name2>=<value2> ..."

Specifies the job environment. Each environment *value* defines the remote environment. The *value* overrides the remote environment values if there is a collision.

drmaa_v_email - "<email1> <email2> ..."

Specifies e-mail addresses that are used to report the job completion and status.

drmaa_block_email - "{0|1}"

Specifies whether e-mail sending shall blocked or not. By default email is not sent. If, however, a setting in a cluster or user settings file or the 'drmaa_native_specification' or 'drmaa_job_category' attribute enables sending email in association with job events, the 'drmaa_block_email' attribute will override that setting, causing no email to be sent.

drmaa_start_time - "[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]"

Specifies the earliest time when the job may be eligible to be run where

```
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
```

```
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
```

If the optional UTC-offset is not specified, the offset associated with the local timezone will be used. If the day (DD) is not specified, the current day will be used unless the specified hour:mm:ss has already elapsed, in which case the next day will be used. Similarly for month (MM), year (YY), and century-1 (CC). Example: The time: Sep 3 4:47:27 PM PDT 2002, could be represented as: 2002/09/03 16:47:27 -07:00.

drmaa_transfer_files - "[i][o][e]"

Specifies, which of the standard I/O files (stdin, stdout and stderr) are to be transferred to/from the execution host. If not set, defaults to "". Any combination of 'e', 'i' and 'o' may be specified. See drmaa_input_path, drmaa_output_path and drmaa_error_path for information about how to specify the standard input file, standard output file and standard error file. The file transfer mechanism itself must be configured by the administrator (see *sge_conf*(5)). When it is configured, the administrator has to enable drmaa_transfer_files. If it is not configured,"drmaa_transfer_files" is not enabled and can't be used.

ENVIRONMENTAL VARIABLES

- *SGE_DRMAA_ALLOW_CWD* Enables the parsing of the -cwd option from the sge_request file(s), job category, and/or the native specification attribute. This option is disabled by default because the algorithm for parsing the -cwd option is not thread-safe.
- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_next_attr_name() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_jobtemplate(3) and *drmaa_submit*(3).

COPYRIGHT

drmaa_init

NAME

drmaa_init, drmaa_exit - Start/finish Altair Grid Engine DRMAA session

SYNOPSIS

```
#include "drmaa.h"

int drmaa_init(
        const char *contact,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_exit(
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

drmaa_init()

The drmaa_init() function initializes the Altair Grid Engine DRMAA API library for all threads of the process and creates a new DRMAA session. This routine must be called once before any other DRMAA call, except for *drmaa_version(3)*, *drmaa_get_DRM_system(3)*, and *drmaa_get_DRMAA_implementation(3)*. Except for the above listed functions, no DRMAA functions may be called before the drmaa_init() function *completes*. Any DRMAA function which is called before the drmaa_init() function completes will return a DRMAA_ERRNO_NO_ACTIVE_SESSION error. *Contact* is an implementation dependent string which may be used to specify which Altair Grid Engine cell to use. If *contact* is NULL, the default Altair Grid Engine cell will be used. In the 1.0 implementation *contact* may have the following value: *session=<session_id>*. To determine the session_id, the *drmaa_get_contact(3)* function should be called after the session has already been initialized. By passing the *session=<session_id>* string to the drmaa_init() function, instead of creating a new session, DRMAA will attempt to reconnect to the session indicated by the *session_id*. The result of reconnecting to a previous session is that all jobs previously submitted in that session *that_are_still_running* will be available in the DRMAA session. Note, however,that jobs which ended before the call to drmaa_init() may not be available or may have no associated exit information or resource usage data.

drmaa_exit()

The drmaa_exit() function closes the DRMAA session for all threads and must be called before process termination. The drmaa_exit() function may be called only once by a single thread in the process and may only be called after the drmaa_init() function has completed. Any DRMAA function, other than *drmaa_init(3)*,which is called after the drmaa_exit() function completes will return a DRMAA_ERRNO_NO_ACTIVE_SESSION error.

The drmaa_exit() function does necessary clean up of the DRMAA session state, including unregistering from the sge_qmaster(8). If the drmaa_exit() function is not called, the qmaster will store events for the DRMAA client until the connection times out, causing extra work for the qmaster and consuming system resources.

Submitted jobs are not affected by the drmaa_exit() function.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

he name of the default cell, i.e. *default*.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which the sge_qmaster is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_init() and drmaa_exit() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_init() and drmaa_exit() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.
The drmaa_init() will fail if:

DRMAA_ERRNO_INVALID_CONTACT_STRING

Initialization failed due to invalid contact string.

DRMAA_ERRNO_DEFAULT_CONTACT_STRING_ERROR

Could not use the default contact string to connect to DRM system.

DRMAA_ERRNO_DRMS_INIT_FAILED

Initialization failed due to failure to init DRM system.

DRMAA_ERRNO_ALREADY_ACTIVE_SESSION

Initialization failed due to existing DRMAA session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_exit() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_DRMS_EXIT_ERROR

DRM system disengagement failed.

SEE ALSO

drmaa_submit(3).

COPYRIGHT

drmaa_job_ps

NAME

drmaa_job_ps, drmaa_control, - Monitor and control jobs

SYNOPSIS

```
#include "drmaa.h"

int drmaa_job_ps(
        const char *job_id,
        int *remote_ps,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_control(
        const char *jobid,
        int action,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_job_ps() function returns the status of the Altair Grid Engine job *job_id* into the integer pointed to by *remote_ps*. Possible return values are

```
DRMAA_PS_UNDETERMINED         job status cannot be determined
DRMAA_PS_QUEUED_ACTIVE        job is queued and active
DRMAA_PS_SYSTEM_ON_HOLD       job is queued and in system hold
DRMAA_PS_USER_ON_HOLD         job is queued and in user hold
DRMAA_PS_USER_SYSTEM_ON_HOLD job is queued and in user and system hold
DRMAA_PS_RUNNING              job is running
DRMAA_PS_SYSTEM_SUSPENDED     job is system suspended
DRMAA_PS_USER_SUSPENDED       job is user suspended
DRMAA_PS_DONE                 job finished normally
DRMAA_PS_FAILED               job finished, but failed
```

Jobs' user hold and user suspend states can be controlled via *drmaa_control(3)*. For affecting system hold and system suspend states the appropriate Altair Grid Engine interfaces must be used.

drmaa_control()

The drmaa_control() function applies control operations on Altair Grid Engine jobs. *jobid* may contain either an Altair Grid Engine jobid or 'DRMAA_JOB_IDS_SESSION_ALL' to refer to all jobs submitted during the DRMAA session opened using drmaa_init(3). Legal values for *action* and their meanings are:

```
DRMAA_CONTROL_SUSPEND        suspend the job
DRMAA_CONTROL_RESUME         resume the job
DRMAA_CONTROL_HOLD           put the job on-hold
DRMAA_CONTROL_RELEASE        release the hold on the job
DRMAA_CONTROL_TERMINATE      kill the job
```

The DRMAA suspend/resume operations are equivalent to the use of `-sj <jobid>' and-usj ' options with Altair Grid Engine qmod(1). The DRMAA hold/release operations are equivalent to the use of Altair Grid Engine qhold(1) and qrls(1). The DRMAA terminate operation is equivalent to the use of Altair Grid Engine qdel(1). Only user hold and user suspend can be controlled via drmaa_control(3). For affecting system hold and system suspend states the appropriate Altair Grid Engine interfaces must be used.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster(8)* is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_job_ps(), and drmaa_control() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer, *error_diagnosis*.

ERRORS

The drmaa_job_ps(), and drmaa_control() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request was not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

DRMAA_ERRNO_INVALID_JOB

The specified job does not exist.
The drmaa_control() will fail if:

DRMAA_ERRNO_RESUME_INCONSISTENT_STATE

The job is not suspended. The resume request will not be processed.

DRMAA_ERRNO_SUSPEND_INCONSISTENT_STATE

The job is not running and thus cannot be suspended.

DRMAA_ERRNO_HOLD_INCONSISTENT_STATE

The job cannot be moved to a hold state.

DRMAA_ERRNO_RELEASE_INCONSISTENT_STATE

The job is not in a hold state.

SEE ALSO

drmaa_submit(3) drmaa_wait(3).

COPYRIGHT

drmaa_jobcontrol

NAME

drmaa_job_ps, drmaa_control, - Monitor and control jobs

SYNOPSIS

```
#include "drmaa.h"

int drmaa_job_ps(
        const char *job_id,
        int *remote_ps,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_control(
        const char *jobid,
        int action,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_job_ps() function returns the status of the Altair Grid Engine job *job_id* into the integer pointed to by *remote_ps*. Possible return values are

```
DRMAA_PS_UNDETERMINED         job status cannot be determined
DRMAA_PS_QUEUED_ACTIVE        job is queued and active
DRMAA_PS_SYSTEM_ON_HOLD       job is queued and in system hold
DRMAA_PS_USER_ON_HOLD         job is queued and in user hold
DRMAA_PS_USER_SYSTEM_ON_HOLD job is queued and in user and system hold
DRMAA_PS_RUNNING              job is running
DRMAA_PS_SYSTEM_SUSPENDED     job is system suspended
DRMAA_PS_USER_SUSPENDED       job is user suspended
DRMAA_PS_DONE                 job finished normally
DRMAA_PS_FAILED               job finished, but failed
```

Jobs' user hold and user suspend states can be controlled via *drmaa_control(3)*. For affecting system hold and system suspend states the appropriate Altair Grid Engine interfaces must be used.

drmaa_control()

The drmaa_control() function applies control operations on Altair Grid Engine jobs. *jobid* may contain either an Altair Grid Engine jobid or 'DRMAA_JOB_IDS_SESSION_ALL' to refer to all jobs submitted during the DRMAA session opened using drmaa_init(3). Legal values for *action* and their meanings are:

```
DRMAA_CONTROL_SUSPEND        suspend the job
DRMAA_CONTROL_RESUME         resume the job
DRMAA_CONTROL_HOLD           put the job on-hold
DRMAA_CONTROL_RELEASE        release the hold on the job
DRMAA_CONTROL_TERMINATE      kill the job
```

The DRMAA suspend/resume operations are equivalent to the use of `-sj <jobid>' and-usj ' options with Altair Grid Engine qmod(1). The DRMAA hold/release operations are equivalent to the use of Altair Grid Engine qhold(1) and qrls(1). The DRMAA terminate operation is equivalent to the use of Altair Grid Engine qdel(1). Only user hold and user suspend can be controlled via drmaa_control(3). For affecting system hold and system suspend states the appropriate Altair Grid Engine interfaces must be used.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster(8)* is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_job_ps(), and drmaa_control() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer, *error_diagnosis*.

ERRORS

The drmaa_job_ps(), and drmaa_control() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request was not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

DRMAA_ERRNO_INVALID_JOB

The specified job does not exist.
The drmaa_control() will fail if:

DRMAA_ERRNO_RESUME_INCONSISTENT_STATE

The job is not suspended. The resume request will not be processed.

DRMAA_ERRNO_SUSPEND_INCONSISTENT_STATE

The job is not running and thus cannot be suspended.

DRMAA_ERRNO_HOLD_INCONSISTENT_STATE

The job cannot be moved to a hold state.

DRMAA_ERRNO_RELEASE_INCONSISTENT_STATE

The job is not in a hold state.

SEE ALSO

drmaa_submit(3) drmaa_wait(3).

COPYRIGHT

drmaa_jobtemplate

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_misc

NAME

drmaa_strerror, drmaa_get_contact, drmaa_version, drmaa_get_DRM_system - Miscellaneous DRMAA functions.

SYNOPSIS

```
#include "drmaa.h"

const char *drmaa_strerror(
        int drmaa_errno
);

int drmaa_get_contact(
        char *contact,
        size_t contact_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_version(
        unsigned int *major,
        unsigned int *minor,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRM_system(
        char *drm_system,
        size_t drm_system_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRMAA_implementation(
        char *drm_impl,
        size_t drm_impl_len,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_strerror() function returns a message text associated with the DRMAA error number,*drmaa_errno*. For invalid DRMAA error codes 'NULL' is returned.

drmaa_get_contact()

The drmaa_get_contact() returns an opaque string containing contact information related to the current DRMAA session to be used with the *drmaa_init*(3) function. The opaque string contains the information required by drmaa_init() to reconnect to the current session instead of creating a new session. *drmaa_init*(3) function.

The drmaa_get_contact() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_version()

The drmaa_version() function returns into the integers pointed to by *major* and *minor*, the major and minor version numbers of the DRMAA library. For a DRMAA 1.0 compliant implementation '1' and '0' will be returned in *major* and *minor*, respectively.

drmaa_get_DRM_system()

The drmaa_get_DRM_system() function returns into *drm_system* up to *drm_system_len* characters of a string containing Altair Grid Engine product and version information.

The drmaa_get_DRM_system() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_get_DRMAA_implementation()

The drmaa_get_DRMAA_implementation() function returns into *drm_system* up to *drm_system_len* characters of a string containing the Altair Grid Engine DRMAA implementation version information. In the current implementation, the drmaa_get_DRMAA_implementation() function returns the same result as the drmaa_get_DRM_system() function.

The drmaa_get_DRMAA_implementation() function returns the same value before and after *drmaa_init*(3) is called.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_contact(), drmaa_version(), and drmaa_get_DRM_system() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_contact(), drmaa_version(), drmaa_get_DRM_system(), and drmaa_get_DRMAA_implementation will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_contact() and drmaa_get_DRM_system() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

SEE ALSO

drmaa_session(3).

COPYRIGHT

drmaa_release_attr_names

NAME

drmaa_get_attribute_names, drmaa_get_vector_attribute_names, drmaa_get_next_attr_name, drmaa_get_num_attr_names, drmaa_release_attr_names - DRMAA job template attributes

SYNOPSIS

```
#include "drmaa.h"

int drmaa_get_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute_names(
        drmaa_attr_names_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_name(
        drmaa_attr_names_t* values,
        char *value,
        int value_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_names(
        drmaa_attr_names_t* values,
        int *size
);

void drmaa_release_attr_names(
        drmaa_attr_names_t* values
);
```

DESCRIPTION

The drmaa_get_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported non-vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The number of names in the names string vector can be determined using *drmaa_get_num_attr_names*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_attribute*(3) and *drmaa_get_attribute*(3) for setting and inspecting non-vector attributes.

drmaa_get_vector_attribute_names()

The drmaa_get_vector_attribute_names() function returns into *values* a DRMAA names string vector containing the set of supported vector DRMAA job template attribute names. The set includes supported DRMAA reserved attribute names and Altair Grid Engine native attribute names. The names in the names string vector can be extracted using *drmaa_get_next_attr_name*(3). The caller is responsible for releasing the names string vector returned into *values* using *drmaa_release_attr_names*(3). Use *drmaa_set_vector_attribute*(3) and *drmaa_get_vector_attribute*(3) for setting and inspecting vector attributes.

drmaa_get_next_attr_name()

Each time drmaa_get_next_attr_name() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA names string vector, *values*. Once the names list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_names()

The drmaa_get_num_attr_names() returns into *size* the number of entries in the DRMAA names string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_names()

The drmaa_release_attr_names() function releases all resources associated with the DRMAA names string vector, *values*.

Attribute Priorities

DRMAA job template attributes can be set from six different sources. In order of precedence, from lowest to highest, these are: options set by DRMAA automatically by default, options set in the *sge_request*(5) file(s), options set in the script file, options set by the drmaa_job_category attribute, options set by the drmaa_native_specification attribute, and options set through other DRMAA attributes.

By default DRMAA sets four options for all jobs. These are “-p 0”, “-b yes”, “-shell no”, and “-w e”. This means that by default, all jobs will have priority 0, all jobs will be treated as binary, i.e. no scripts args will be parsed, all jobs will be executed without a wrapper shell, and jobs which are unschedulable will cause a submit error.

The *sge_request*(5) file, found in the $SGE_ROOT/$SGE_CELL/common directory, may contain options to be applied to all jobs. The .sge_request file found in the user’s home directory or the current working directory may also contain options to be applied to certain jobs. See *sge_request*(5) for more information. If the *sge_request*(5) file contains “-b no” or if the drmaa_native_specification attribute is set and contains “-b no”, the script file will be parsed for in-line arguments. Otherwise, no scripts args will be interpreted. See *qsub*(1) for more information. If the drmaa_job_category attribute is set, and the category it points to exists in one of the *qtask* files, the options associated with that category will be applied to the job template.

If the drmaa_native_specification attribute is set, all options contained therein will be applied to the job template. See the drmaa_native_specification below for more information. Other DRMAA attributes will override any previous settings. For example, if the sge_request file contains “-j y”, but the drmaa_join_files attribute is set to “n”, the ultimate result is that the input and output files will remain separate.

For various reasons, some options are silently ignored by DRMAA. Setting any of these options will have no effect. The ignored options are: -cwd, -help, -sync, -t, -verify, -w w, and -w v. The -cwd option can be re-enabled by setting the environment variable, SGE_DRMAA_ALLOW_CWD. However, the -cwd option is not thread safe and should not be used in a multi-threaded context.

Attribute Correlations

The following DRMAA attributes correspond to the following *qsub*(1) options:

```
DRMAA Attribute                  qsub Option
-----------------------------------------------------
drmaa_remote_command             script file
drmaa_v_argv                     script file args
drmaa_js_state = "drmaa_hold"    -h
drmaa_v_env                      -v
drmaa_wd = \$PWD                  -cwd
drmaa_job_category               (qtsch qtask)*
drmaa_native_specification       ALL*
drmaa_v_email                    -M
drmaa_block_email = "1"          -m n
drmaa_start_time                 -a
drmaa_job_name                   -N
drmaa_input_path                 -i
drmaa_output_path                -o
drmaa_error_path                 -e
drmaa_join_files                 -j
drmaa_transfer_files             (prolog and epilog)*
```

```
* See the individual attribute description below
```

DRMAA JOB TEMPLATE ATTRIBUTES

drmaa_remote_command - "<remote_command>"

Specifies the remote command to execute. The *remote_command* must be the path of an executable that is available at the job's execution host. If the path is relative, it is assumed to be relative to the working directory, usually set through the drmaa_wd attribute. If working directory is not set, the path is assumed to be relative to the user's home directory.

The file pointed to by remote_command may either be an executable binary or an executable script. If a script, it must include the path to the shell in a #! line at the beginning of the script. By default, the remote command will be executed directly, as by *exec*(2). To have the remote command executed in a shell, such as to preserve environment settings, use the drmaa_native_specification attribute to include the "-shell yes" option. Jobs which are executed by a wrapper shell fail differently from jobs which are executed directly. When a job which contains a user error, such as an invalid path to the executable, is executed by a wrapper shell, the job will execute successfully, but exit with a return code of 1. When a job which contains such an error is executed directly, it will enter the DRMAA_PS_FAILED state upon execution.

drmaa_js_state - "{drmaa_hold|drmaa_active}"

Specifies the job state at submission. The string values 'drmaa_hold' and 'drmaa_active' are supported. When 'drmaa_active' is used the job is submitted in a runnable state. When 'drmaa_hold' is used the job is submitted in user hold state (either DRMAA_PS_USER_ON_HOLD or DRMAA_PS_USER_SYSTEM_ON_HOLD). This attribute is largely equivalent to the *qsub*(1) submit option '-h'.

drmaa_wd - "<directory_name>"

Specifies the directory name where the job will be executed. A '$drmaa_hd_ph$' placeholder at the beginning of the *directory_name* denotes the remaining string portion as a relative directory name that is resolved relative to the job user's home directory at the execution host. When the DRMAA job template is used for bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *directory_name* to cause a substitution with the parametric job's index. The *directory_name* must be specified in a syntax that is common at the host where the job is executed. If set to a relative path and no placeholder is used, a path relative to the user's home directory is assumed. If not set, the working directory will default to the user's home directory. If set and the given directory does not exist the job will enter the DRMAA_PS_FAILED state when run.

Note that the working directory path is the path on the execution host. If binary mode is disabled, an attempt to find the job script will be made, relative to the working directory

path. That means that the path to the script must be the same on both the submission and execution hosts.

drmaa_job_name - "<job_name>"

Specifies the job's name. Setting the job name is equivalent to use of *qsub*(1) submit option '-N' with *job_name* as option argument.

drmaa_input_path - "[<hostname>]:<file_path>"

Specifies the standard input of the job. Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream. If the standard input is set it specifies the network path of the job's input stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'i', the input file will be fetched by Altair Grid Engine from the specified host or from the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for bulk job submission, (See also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be read the job enters the state DRMAA_PS_FAILED.

drmaa_output_path - "[<hostname>]:<file_path>"

Specifies the standard output of the job. If not explicitly set in the job template, the whereabouts of the job's output stream is not defined. If set, this attribute specifies the network path of the job's output stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'o', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the

file_path denotes the remaining portion of *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.

drmaa_error_path - "[<hostname>]:<file_path>"

Specifies the standard error of the job. If not explicitly set in the job template, the whereabouts of the job's error stream is not defined. If set, this attribute specifies the network path of the job's error stream file.

When the 'drmaa_transfer_files' job template attribute is supported and contains the character 'e', the output file will be transferred by Altair Grid Engine to the specified host or to the submit host if no *hostname* is specified. When the 'drmaa_transfer_files' job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed regardless of any *hostname* specified.

If the DRMAA job template will be used for of bulk job submission (see also *drmaa_run_bulk_jobs*(3)) the '$drmaa_incr_ph$' placeholder can be used at any position within the *file_path* to cause a substitution with the parametric job's index. A '$drmaa_hd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job user's home directory at the host where the file is located. A '$drmaa_wd_ph$' placeholder at the beginning of the *file_path* denotes the remaining portion of the *file_path* as a relative file specification resolved relative to the job's working directory at the host where the file is located. The *file_path* must be specified in a syntax that is common at the host where the file is located. If set and the file can't be written before execution the job enters the state DRMAA_PS_FAILED. The attribute name is drmaa_error_path.

drmaa_join_files - "{y|n}"

Specifies if the job's error stream should be intermixed with the output stream. If not explicitly set in the job template the attribute defaults to 'n'. Either 'y' or 'n' can be specified. If 'y' is specified Altair Grid Engine will ignore the value of the 'drmaa_error_path' job template attribute and intermix the standard error stream with the standard output stream as specified with 'drmaa_output_path'.

drmaa_v_argv - "argv1 argv2 . . ."

Specifies the arguments to the job.

drmaa_job_category - "<category>"

Specifies the DRMAA job category. The *category* string is used by Altair Grid Engine as a reference into the *qtask* file. Certain *qsub*(1) options used in the referenced qtask file line are applied to the job template before submission to allow site-specific resolving of resources and/or policies. The cluster qtask file, the local qtask file, and the user qtask file

are searched. Job settings resulting from job template category are overridden by settings resulting from the job template drmaa_native_specification attribute as well as by explicit DRMAA job template settings.

In order to avoid collisions with command names in the qtask files, it is recommended that DRMAA job category names take the form: <category_name>.cat.

The options -help, -sync, -t, -verify, and -w w|v are ignored. The -cwd option is ignored unless the $SGE_DRMAA_ALLOW_CWD environment variable is set.

drmaa_native_specification - "<native_specification>"

Specifies Altair Grid Engine native *qsub*(1) options which will be interpreted as part of the DRMAA job template. All options available to *qsub*(1) command may be used in the *native_specification*, except for -help, -sync, -t, -verify, and -w w|v. The -cwd option may only be used if the SGE_DRMAA_ALLOW_CWD environment variable is set. This is because the current parsing algorithm for -cwd is not thread-safe. Options set in the *native specification* will be overridden by the corresponding DRMAA attributes. See *qsub*(1) for more information on qsub options.

drmaa_v_env - "<name1>=<value1> <name2>=<value2> ..."

Specifies the job environment. Each environment *value* defines the remote environment. The *value* overrides the remote environment values if there is a collision.

drmaa_v_email - "<email1> <email2> ..."

Specifies e-mail addresses that are used to report the job completion and status.

drmaa_block_email - "{0|1}"

Specifies whether e-mail sending shall blocked or not. By default email is not sent. If, however, a setting in a cluster or user settings file or the 'drmaa_native_specification' or 'drmaa_job_category' attribute enables sending email in association with job events, the 'drmaa_block_email' attribute will override that setting, causing no email to be sent.

drmaa_start_time - "[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]"

Specifies the earliest time when the job may be eligible to be run where

```
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
```

```
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
```

If the optional UTC-offset is not specified, the offset associated with the local timezone will be used. If the day (DD) is not specified, the current day will be used unless the specified hour:mm:ss has already elapsed, in which case the next day will be used. Similarly for month (MM), year (YY), and century-1 (CC). Example: The time: Sep 3 4:47:27 PM PDT 2002, could be represented as: 2002/09/03 16:47:27 -07:00.

drmaa_transfer_files - "[i][o][e]"

Specifies, which of the standard I/O files (stdin, stdout and stderr) are to be transferred to/from the execution host. If not set, defaults to "". Any combination of 'e', 'i' and 'o' may be specified. See drmaa_input_path, drmaa_output_path and drmaa_error_path for information about how to specify the standard input file, standard output file and standard error file. The file transfer mechanism itself must be configured by the administrator (see *sge_conf*(5)). When it is configured, the administrator has to enable drmaa_transfer_files. If it is not configured,"drmaa_transfer_files" is not enabled and can't be used.

ENVIRONMENTAL VARIABLES

- *SGE_DRMAA_ALLOW_CWD* Enables the parsing of the -cwd option from the sge_request file(s), job category, and/or the native specification attribute. This option is disabled by default because the algorithm for parsing the -cwd option is not thread-safe.
- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_attribute_names(), drmaa_get_vector_attribute_names(), and drmaa_get_next_attr_name() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_next_attr_name() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_jobtemplate(3) and *drmaa_submit*(3).

COPYRIGHT

drmaa_release_attr_values

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_release_job_ids

NAME

drmaa_run_job, drmaa_run_bulk_jobs, drmaa_get_next_job_id, drmaa_get_num_job_ids, drmaa_release_job_ids - Job submission

SYNOPSIS

```
#include "drmaa.h"

int drmaa_run_job(
        char *job_id,
        size_t job_id_len,
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_run_bulk_jobs(
        drmaa_job_ids_t **jobids,
        drmaa_job_template_t *jt,
        int start,
        int end,
        int incr,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_job_id(
        drmaa_job_ids_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_job_ids(
        drmaa_job_ids_t* values,
        int *size
);

void drmaa_release_job_ids(
        drmaa_job_ids_t* values
);
```

DESCRIPTION

drmaa_run_job() submits an xAltair Grid Engine job with attributes defined in the DRMAA job template, *jt*. On success up to *job_id_len* bytes of the job identifier are returned into the buffer, *job_id*.

drmaa_run_bulk_jobs()

The drmaa_run_bulk_jobs() submits a Altair Grid Engine array job very much as if the *qsub*(1) option '-t *start*-*end*:*incr*' had been used along with the additional attributes defined in the DRMAA job template, *jt*. The same constraints regarding value ranges are also in effect for the parameters *start*, *end*, and *incr* as for *qsub*(1) -t. On success a DRMAA job id string vector containing job identifiers for each array job task is returned into *jobids*. The job identifiers in the job id string vector can be extracted using *drmaa_get_next_job_id*(3). The number of identifiers in the job id string vector can be determined using *drmaa_get_num_job_ids*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the job id string vector returned into *jobids* using *drmaa_release_job_ids*(3).

drmaa_get_next_job_id()

Each time drmaa_get_next_job_id() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA job id string vector, *values*. Once the job ids list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_job_ids()

The drmaa_get_num_job_ids() returns into *size* the number of entries in the DRMAA job ids string vector. This function is only available in the 1.0 implementation.

drmaa_release_job_ids()

The drmaa_release_attr_job_id() function releases all resources associated with the DRMAA job id string vector, *values*. This operation has no effect on the actual xAltair Grid Engine array job tasks.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default xAltair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_run_job() and drmaa_run_bulk_jobs() functions will fail if:

DRMAA_ERRNO_TRY_LATER

The DRM system indicated that it is too busy to accept the job. A retry may succeed, however.

DRMAA_ERRNO_DENIED_BY_DRM

The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

The drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

ENVIRONMENTAL VARIABLES

SGE_DRMAA_ENABLE_ERROR_STATE

When this environment variable is set, then jobs that are submitted with drmaa_run_job() or drmaa_run_bulk_jobs() will change into error state when either during the job start or during the execution of the job an error occurs. Normally DRMAA jobs will not switch into error state when something fails.

SEE ALSO

drmaa_attributes(3), *drmaa_jobtemplate*(3).

COPYRIGHT

drmaa_run_bulk_jobs

NAME

drmaa_run_job, drmaa_run_bulk_jobs, drmaa_get_next_job_id, drmaa_get_num_job_ids, drmaa_release_job_ids - Job submission

SYNOPSIS

```
#include "drmaa.h"

int drmaa_run_job(
        char *job_id,
        size_t job_id_len,
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_run_bulk_jobs(
        drmaa_job_ids_t **jobids,
        drmaa_job_template_t *jt,
        int start,
        int end,
        int incr,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_job_id(
        drmaa_job_ids_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_job_ids(
        drmaa_job_ids_t* values,
        int *size
);

void drmaa_release_job_ids(
        drmaa_job_ids_t* values
);
```

DESCRIPTION

drmaa_run_job() submits an xAltair Grid Engine job with attributes defined in the DRMAA job template, *jt*. On success up to *job_id_len* bytes of the job identifier are returned into the buffer, *job_id*.

drmaa_run_bulk_jobs()

The drmaa_run_bulk_jobs() submits a Altair Grid Engine array job very much as if the *qsub*(1) option '-t *start*-*end*:*incr*' had been used along with the additional attributes defined in the DRMAA job template, *jt*. The same constraints regarding value ranges are also in effect for the parameters *start*, *end*, and *incr* as for *qsub*(1) -t. On success a DRMAA job id string vector containing job identifiers for each array job task is returned into *jobids*. The job identifiers in the job id string vector can be extracted using *drmaa_get_next_job_id*(3). The number of identifiers in the job id string vector can be determined using *drmaa_get_num_job_ids*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the job id string vector returned into *jobids* using *drmaa_release_job_ids*(3).

drmaa_get_next_job_id()

Each time drmaa_get_next_job_id() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA job id string vector, *values*. Once the job ids list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_job_ids()

The drmaa_get_num_job_ids() returns into *size* the number of entries in the DRMAA job ids string vector. This function is only available in the 1.0 implementation.

drmaa_release_job_ids()

The drmaa_release_attr_job_id() function releases all resources associated with the DRMAA job id string vector, *values*. This operation has no effect on the actual xAltair Grid Engine array job tasks.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default xAltair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_run_job() and drmaa_run_bulk_jobs() functions will fail if:

DRMAA_ERRNO_TRY_LATER

The DRM system indicated that it is too busy to accept the job. A retry may succeed, however.

DRMAA_ERRNO_DENIED_BY_DRM

The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

The drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

ENVIRONMENTAL VARIABLES

SGE_DRMAA_ENABLE_ERROR_STATE

When this environment variable is set, then jobs that are submitted with drmaa_run_job() or drmaa_run_bulk_jobs() will change into error state when either during the job start or during the execution of the job an error occurs. Normally DRMAA jobs will not switch into error state when something fails.

SEE ALSO

drmaa_attributes(3), *drmaa_jobtemplate*(3).

COPYRIGHT

drmaa_run_job

NAME

drmaa_run_job, drmaa_run_bulk_jobs, drmaa_get_next_job_id, drmaa_get_num_job_ids, drmaa_release_job_ids - Job submission

SYNOPSIS

```
#include "drmaa.h"

int drmaa_run_job(
        char *job_id,
        size_t job_id_len,
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_run_bulk_jobs(
        drmaa_job_ids_t **jobids,
        drmaa_job_template_t *jt,
        int start,
        int end,
        int incr,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_job_id(
        drmaa_job_ids_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_job_ids(
        drmaa_job_ids_t* values,
        int *size
);

void drmaa_release_job_ids(
        drmaa_job_ids_t* values
);
```

DESCRIPTION

drmaa_run_job() submits an xAltair Grid Engine job with attributes defined in the DRMAA job template, *jt*. On success up to *job_id_len* bytes of the job identifier are returned into the buffer, *job_id*.

drmaa_run_bulk_jobs()

The drmaa_run_bulk_jobs() submits a Altair Grid Engine array job very much as if the *qsub*(1) option '-t *start*-*end*:*incr*' had been used along with the additional attributes defined in the DRMAA job template, *jt*. The same constraints regarding value ranges are also in effect for the parameters *start*, *end*, and *incr* as for *qsub*(1) -t. On success a DRMAA job id string vector containing job identifiers for each array job task is returned into *jobids*. The job identifiers in the job id string vector can be extracted using *drmaa_get_next_job_id*(3). The number of identifiers in the job id string vector can be determined using *drmaa_get_num_job_ids*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the job id string vector returned into *jobids* using *drmaa_release_job_ids*(3).

drmaa_get_next_job_id()

Each time drmaa_get_next_job_id() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA job id string vector, *values*. Once the job ids list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_job_ids()

The drmaa_get_num_job_ids() returns into *size* the number of entries in the DRMAA job ids string vector. This function is only available in the 1.0 implementation.

drmaa_release_job_ids()

The drmaa_release_attr_job_id() function releases all resources associated with the DRMAA job id string vector, *values*. This operation has no effect on the actual xAltair Grid Engine array job tasks.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default xAltair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_run_job() and drmaa_run_bulk_jobs() functions will fail if:

DRMAA_ERRNO_TRY_LATER

The DRM system indicated that it is too busy to accept the job. A retry may succeed, however.

DRMAA_ERRNO_DENIED_BY_DRM

The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

The drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

ENVIRONMENTAL VARIABLES

SGE_DRMAA_ENABLE_ERROR_STATE

When this environment variable is set, then jobs that are submitted with drmaa_run_job() or drmaa_run_bulk_jobs() will change into error state when either during the job start or during the execution of the job an error occurs. Normally DRMAA jobs will not switch into error state when something fails.

SEE ALSO

drmaa_attributes(3), *drmaa_jobtemplate*(3).

COPYRIGHT

drmaa_session

NAME

drmaa_init, drmaa_exit - Start/finish Altair Grid Engine DRMAA session

SYNOPSIS

```
#include "drmaa.h"

int drmaa_init(
        const char *contact,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_exit(
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

drmaa_init()

The drmaa_init() function initializes the Altair Grid Engine DRMAA API library for all threads of the process and creates a new DRMAA session. This routine must be called once before any other DRMAA call, except for *drmaa_version(3)*, *drmaa_get_DRM_system(3)*, and *drmaa_get_DRMAA_implementation(3)*. Except for the above listed functions, no DRMAA functions may be called before the drmaa_init() function *completes*. Any DRMAA function which is called before the drmaa_init() function completes will return a DRMAA_ERRNO_NO_ACTIVE_SESSION error. *Contact* is an implementation dependent string which may be used to specify which Altair Grid Engine cell to use. If *contact* is NULL, the default Altair Grid Engine cell will be used. In the 1.0 implementation *contact* may have the following value: *session=<session_id>*. To determine the session_id, the *drmaa_get_contact(3)* function should be called after the session has already been initialized. By passing the *session=<session_id>* string to the drmaa_init() function, instead of creating a new session, DRMAA will attempt to reconnect to the session indicated by the *session_id*. The result of reconnecting to a previous session is that all jobs previously submitted in that session *that_are_still_running* will be available in the DRMAA session. Note, however,that jobs which ended before the call to drmaa_init() may not be available or may have no associated exit information or resource usage data.

drmaa_exit()

The drmaa_exit() function closes the DRMAA session for all threads and must be called before process termination. The drmaa_exit() function may be called only once by a single thread in the process and may only be called after the drmaa_init() function has completed. Any DRMAA function, other than *drmaa_init(3)*,which is called after the drmaa_exit() function completes will return a DRMAA_ERRNO_NO_ACTIVE_SESSION error.

The drmaa_exit() function does necessary clean up of the DRMAA session state, including unregistering from the sge_qmaster(8). If the drmaa_exit() function is not called, the qmaster will store events for the DRMAA client until the connection times out, causing extra work for the qmaster and consuming system resources.

Submitted jobs are not affected by the drmaa_exit() function.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

he name of the default cell, i.e. *default*.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which the sge_qmaster is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_init() and drmaa_exit() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_init() and drmaa_exit() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.
The drmaa_init() will fail if:

DRMAA_ERRNO_INVALID_CONTACT_STRING

Initialization failed due to invalid contact string.

DRMAA_ERRNO_DEFAULT_CONTACT_STRING_ERROR

Could not use the default contact string to connect to DRM system.

DRMAA_ERRNO_DRMS_INIT_FAILED

Initialization failed due to failure to init DRM system.

DRMAA_ERRNO_ALREADY_ACTIVE_SESSION

Initialization failed due to existing DRMAA session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_exit() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_DRMS_EXIT_ERROR

DRM system disengagement failed.

SEE ALSO

drmaa_submit(3).

COPYRIGHT

drmaa_set_attribute

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_set_vector_attribute

NAME

drmaa_allocate_job_template, drmaa_delete_job_template, drmaa_set_attribute, drmaa_get_attribute, drmaa_set_vector_attribute, drmaa_get_vector_attribute, drmaa_get_next_attr_value, drmaa_get_num_attr_values, drmaa_release_attr_values - Altair Grid Engine DRMAA job template handling

SYNOPSIS

```
#include "drmaa.h"

int drmaa_allocate_job_template(
        drmaa_job_template_t **jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_delete_job_template(
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        size_t value_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_set_vector_attribute(
```

```
        drmaa_job_template_t *jt,
        const char *name,
        const char *value[],
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_vector_attribute(
        drmaa_job_template_t *jt,
        const char *name,
        drmaa_attr_values_t **values,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_attr_value(
        drmaa_attr_values_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_attr_values(
        drmaa_attr_values_t* values,
        int *size
);

void drmaa_release_attr_values(
        drmaa_attr_values_t* values
);
```

DESCRIPTION

The drmaa_allocate_job_template() function allocates a new DRMAA job template into *jt*. DRMAA job templates describe specifics of jobs that are submitted using drmaa_run_job(3) and drmaa_run_bulk_jobs(3).

drmaa_delete_job_template()

The drmaa_delete_job_template() function releases all resources associated with the DRMAA job template *jt*. Jobs that were submitted using the job template are not affected.

drmaa_set_attribute()

The drmaa_set_attribute() function stores the *value* under *name* for the given DRMAA job template, *jt* . Only non-vector attributes may be passed.

drmaa_get_attribute()

The drmaa_get_attribute() function returns into *value* up to *value_len* bytes from the string stored for the non-vector attribute, *name*, in the DRMAA job template, *jt*.

drmaa_set_vector_attribute()

The drmaa_set_vector_attribute() function stores the strings in *value* under *name* in the list of vector attributes for the given DRMAA job template, *jt* . Only vector attributes may be passed. The *value* pointer array must be *NULL* terminated.

drmaa_get_vector_attribute()

The drmaa_get_vector_attribute() function returns into *values* a DRMAA attribute string vector containing all string values stored in the vector attribute, *name*. The values in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). The caller is responsible for releasing the DRMAA values string vector returned into *values* using *drmaa_release_attr_values*(3).

drmaa_get_next_attr_value()

Each time drmaa_get_next_attr_value() is called it returns into *value* up to *value_len* bytes of the next entry stored in the DRMAA values string vector, *values*. Once the values list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_attr_values()

The drmaa_get_num_attr_values() returns into *size* the number of entries in the DRMAA values string vector. This function is only available in the 1.0 implementation.

drmaa_release_attr_values()

The drmaa_release_attr_values() function releases all resources associated with the DRMAA values string vector, *values*.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_allocate_job_template(), drmaa_delete_job_template(), drmaa_set_attribute(), drmaa_get_attribute(), drmaa_set_vector_attribute(), drmaa_get_vector_attribute(), and drmaa_get_next_attr_value() functions will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.
The drmaa_set_attribute() and drmaa_set_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

The format for the attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The value for the attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

The value of this attribute is conflicting with a previously set attributes.
The drmaa_get_attribute() and drmaa_get_vector_attribute() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

The specified attribute is not set in the DRMAA job template.
The drmaa_get_next_attr_value() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

SEE ALSO

drmaa_submit(3) and *drmaa_attributes*(3).

COPYRIGHT

drmaa_strerror

NAME

drmaa_strerror, drmaa_get_contact, drmaa_version, drmaa_get_DRM_system - Miscellaneous DRMAA functions.

SYNOPSIS

```
#include "drmaa.h"

const char *drmaa_strerror(
        int drmaa_errno
);

int drmaa_get_contact(
        char *contact,
        size_t contact_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_version(
        unsigned int *major,
        unsigned int *minor,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRM_system(
        char *drm_system,
        size_t drm_system_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRMAA_implementation(
        char *drm_impl,
        size_t drm_impl_len,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_strerror() function returns a message text associated with the DRMAA error number,*drmaa_errno*. For invalid DRMAA error codes 'NULL' is returned.

drmaa_get_contact()

The drmaa_get_contact() returns an opaque string containing contact information related to the current DRMAA session to be used with the *drmaa_init*(3) function. The opaque string contains the information required by drmaa_init() to reconnect to the current session instead of creating a new session. *drmaa_init*(3) function.

The drmaa_get_contact() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_version()

The drmaa_version() function returns into the integers pointed to by *major* and *minor*, the major and minor version numbers of the DRMAA library. For a DRMAA 1.0 compliant implementation '1' and '0' will be returned in *major* and *minor*, respectively.

drmaa_get_DRM_system()

The drmaa_get_DRM_system() function returns into *drm_system* up to *drm_system_len* characters of a string containing Altair Grid Engine product and version information.

The drmaa_get_DRM_system() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_get_DRMAA_implementation()

The drmaa_get_DRMAA_implementation() function returns into *drm_system* up to *drm_system_len* characters of a string containing the Altair Grid Engine DRMAA implementation version information. In the current implementation, the drmaa_get_DRMAA_implementation() function returns the same result as the drmaa_get_DRM_system() function.

The drmaa_get_DRMAA_implementation() function returns the same value before and after *drmaa_init*(3) is called.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_contact(), drmaa_version(), and drmaa_get_DRM_system() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_contact(), drmaa_version(), drmaa_get_DRM_system(), and drmaa_get_DRMAA_implementation will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_contact() and drmaa_get_DRM_system() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

SEE ALSO

drmaa_session(3).

COPYRIGHT

drmaa_submit

NAME

drmaa_run_job, drmaa_run_bulk_jobs, drmaa_get_next_job_id, drmaa_get_num_job_ids, drmaa_release_job_ids - Job submission

SYNOPSIS

```
#include "drmaa.h"

int drmaa_run_job(
        char *job_id,
        size_t job_id_len,
        drmaa_job_template_t *jt,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_run_bulk_jobs(
        drmaa_job_ids_t **jobids,
        drmaa_job_template_t *jt,
        int start,
        int end,
        int incr,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_next_job_id(
        drmaa_job_ids_t* values,
        char *value,
        int value_len
);

int drmaa_get_num_job_ids(
        drmaa_job_ids_t* values,
        int *size
);

void drmaa_release_job_ids(
        drmaa_job_ids_t* values
);
```

DESCRIPTION

drmaa_run_job() submits an xAltair Grid Engine job with attributes defined in the DRMAA job template, *jt*. On success up to *job_id_len* bytes of the job identifier are returned into the buffer, *job_id*.

drmaa_run_bulk_jobs()

The drmaa_run_bulk_jobs() submits a Altair Grid Engine array job very much as if the *qsub*(1) option '-t *start*-*end*:*incr*' had been used along with the additional attributes defined in the DRMAA job template, *jt*. The same constraints regarding value ranges are also in effect for the parameters *start*, *end*, and *incr* as for *qsub*(1) -t. On success a DRMAA job id string vector containing job identifiers for each array job task is returned into *jobids*. The job identifiers in the job id string vector can be extracted using *drmaa_get_next_job_id*(3). The number of identifiers in the job id string vector can be determined using *drmaa_get_num_job_ids*(3). Note that this function is only available in the 1.0 implementation. The caller is responsible for releasing the job id string vector returned into *jobids* using *drmaa_release_job_ids*(3).

drmaa_get_next_job_id()

Each time drmaa_get_next_job_id() is called it returns into the buffer, *value*, up to *value_len* bytes of the next entry stored in the DRMAA job id string vector, *values*. Once the job ids list has been exhausted, DRMAA_ERRNO_NO_MORE_ELEMENTS is returned.

drmaa_get_num_job_ids()

The drmaa_get_num_job_ids() returns into *size* the number of entries in the DRMAA job ids string vector. This function is only available in the 1.0 implementation.

drmaa_release_job_ids()

The drmaa_release_attr_job_id() function releases all resources associated with the DRMAA job id string vector, *values*. This operation has no effect on the actual xAltair Grid Engine array job tasks.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default xAltair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_run_job() and drmaa_run_bulk_jobs() functions will fail if:

DRMAA_ERRNO_TRY_LATER

The DRM system indicated that it is too busy to accept the job. A retry may succeed, however.

DRMAA_ERRNO_DENIED_BY_DRM

The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

The drmaa_get_next_job_id() will fail if:

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

When there are no more entries in the vector.

ENVIRONMENTAL VARIABLES

SGE_DRMAA_ENABLE_ERROR_STATE

When this environment variable is set, then jobs that are submitted with drmaa_run_job() or drmaa_run_bulk_jobs() will change into error state when either during the job start or during the execution of the job an error occurs. Normally DRMAA jobs will not switch into error state when something fails.

SEE ALSO

drmaa_attributes(3), *drmaa_jobtemplate*(3).

COPYRIGHT

drmaa_synchronize

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_version

NAME

drmaa_strerror, drmaa_get_contact, drmaa_version, drmaa_get_DRM_system - Miscellaneous DRMAA functions.

SYNOPSIS

```
#include "drmaa.h"

const char *drmaa_strerror(
        int drmaa_errno
);

int drmaa_get_contact(
        char *contact,
        size_t contact_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_version(
        unsigned int *major,
        unsigned int *minor,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRM_system(
        char *drm_system,
        size_t drm_system_len,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_get_DRMAA_implementation(
        char *drm_impl,
        size_t drm_impl_len,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_strerror() function returns a message text associated with the DRMAA error number,*drmaa_errno*. For invalid DRMAA error codes 'NULL' is returned.

drmaa_get_contact()

The drmaa_get_contact() returns an opaque string containing contact information related to the current DRMAA session to be used with the *drmaa_init*(3) function. The opaque string contains the information required by drmaa_init() to reconnect to the current session instead of creating a new session. *drmaa_init*(3) function.

The drmaa_get_contact() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_version()

The drmaa_version() function returns into the integers pointed to by *major* and *minor*, the major and minor version numbers of the DRMAA library. For a DRMAA 1.0 compliant implementation '1' and '0' will be returned in *major* and *minor*, respectively.

drmaa_get_DRM_system()

The drmaa_get_DRM_system() function returns into *drm_system* up to *drm_system_len* characters of a string containing Altair Grid Engine product and version information.

The drmaa_get_DRM_system() function returns the same value before and after *drmaa_init*(3) is called.

drmaa_get_DRMAA_implementation()

The drmaa_get_DRMAA_implementation() function returns into *drm_system* up to *drm_system_len* characters of a string containing the Altair Grid Engine DRMAA implementation version information. In the current implementation, the drmaa_get_DRMAA_implementation() function returns the same result as the drmaa_get_DRM_system() function.

The drmaa_get_DRMAA_implementation() function returns the same value before and after *drmaa_init*(3) is called.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_get_contact(), drmaa_version(), and drmaa_get_DRM_system() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_get_contact(), drmaa_version(), drmaa_get_DRM_system(), and drmaa_get_DRMAA_implementation will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_get_contact() and drmaa_get_DRM_system() will fail if:

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

SEE ALSO

drmaa_session(3).

COPYRIGHT

drmaa_wait

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

```
The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. *default*.
```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_wcoredump

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_wexitstatus

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_wifaborted

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_wifexited

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_wifsignaled

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

drmaa_wtermsig

NAME

drmaa_synchronize, drmaa_wait, drmaa_wifexited, drmaa_wexitstatus, drmaa_wifsignaled, drmaa_wtermsig, drmaa_wcoredump, drmaa_wifaborted - Waiting for jobs to finish

SYNOPSIS

```
#include "drmaa.h"

int drmaa_synchronize(
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wait(
        const char *job_id,
        char *job_id_out,
        size_t job_id_out_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diagnois_len
);

int drmaa_wifaborted(
        int *aborted,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wifexited(
        int *exited,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

```
int drmaa_wifsignaled(
        int *signaled,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wcoredump(
        int *core_dumped,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wexitstatus(
        int *exit_status,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);

int drmaa_wtermsig(
        char *signal,
        size_t signal_len,
        int stat,
        char *error_diagnosis,
        size_t error_diag_len
);
```

DESCRIPTION

The drmaa_synchronize() function blocks the calling thread until all jobs specified in *job_ids* have failed or finished execution. If *job_ids* contains 'DRMAA_JOB_IDS_SESSION_ALL', then this function waits for all jobs submitted during this DRMAA session. The *job_ids* pointer array must be *NULL* terminated.

To prevent blocking indefinitely in this call, the caller may use the *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be used to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The *dispose* parameter specifies how to treat reaping information. If '0' is passed to this parameter, job finish information will still be available when *drmaa_wait*(3) is used. If '1' is passed, *drmaa_wait*(3) will be unable to access this job's finish information.

drmaa_wait()

The drmaa_wait() function blocks the calling thread until a job fails or finishes execution. This routine is modeled on the *wait4*(3) routine. If the special string 'DRMAA_JOB_IDS_SESSION_ANY' is passed as *job_id*, this routine will wait for any job from the session. Otherwise the *job_id* must be the job identifier of a job or array job task that was submitted during the session.

To prevent blocking indefinitely in this call, the caller may use *timeout*, specifying how many seconds to wait for this call to complete before timing out. The special value DRMAA_TIMEOUT_WAIT_FOREVER can be to wait indefinitely for a result. The special value DRMAA_TIMEOUT_NO_WAIT can be used to return immediately. If the call exits before *timeout* seconds have passed, all the specified jobs have completed or the calling thread received an interrupt. In both cases, the return code is DRMAA_ERRNO_EXIT_TIMEOUT.

The routine reaps jobs on a successful call, so any subsequent calls to *drmaa_wait*(3) will fail returning a DRMAA_ERRNO_INVALID_JOB error, meaning that the job has already been reaped. This error is the same as if the job were unknown. Returning due to an elapsed timeout or an interrupt does not cause the job information to be reaped. This means that, in this case, it is possible to issue *drmaa_wait*(3) multiple times for the same *job_id*.

If *job_id_out* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, up to *job_id_out_len* characters from the job id of the failed or finished job are returned.

If *stat* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, the status of the job is stored in the integer pointed to by *stat*. *stat* indicates whether job failed or finished and other information. The information encoded in the integer value can be accessed via *drmaa_wifaborted*(3) *drmaa_wifexited*(3) *drmaa_wifsignaled*(3) *drmaa_wcoredump*(3) *drmaa_wexitstatus*(3) *drmaa_wtermsig*(3).

If *rusage* is not a null pointer, then on return from a successful *drmaa_wait*(3) call, a summary of the resources used by the terminated job is returned in form of a DRMAA values string vector. The entries in the DRMAA values string vector can be extracted using *drmaa_get_next_attr_value*(3). Each string returned by *drmaa_get_next_attr_value*(3) will be of the format <name>=<value>, where <name> and <value> specify name and amount of resources consumed by the job, respectively. See *sge_accounting*(5) for an explanation of the resource information.

drmaa_wifaborted()

The drmaa_wifaborted() function evaluates into the integer pointed to by *aborted* a non-zero value if *stat* was returned from a job that ended before entering the running state.

drmaa_wifexited()

The drmaa_wifexited() function evaluates into the integer pointed to by *exited* a non-zero value if *stat* was returned from a job that terminated normally. A zero value can also indicate that although the job has terminated normally, an exit status is not available, or that it is not known whether the job terminated normally. In both cases *drmaa_wexitstatus*(3) will not provide exit status information. A non-zero value returned in *exited* indicates more

detailed diagnosis can be provided by means of *drmaa_wifsignaled*(3), *drmaa_wtermsig*(3) and *drmaa_wcoredump*(3).

drmaa_wifsignaled()

The drmaa_wifsignaled() function evaluates into the integer pointed to by *signaled* a non-zero value if *stat* was returned for a job that terminated due to the receipt of a signal. A zero value can also indicate that although the job has terminated due to the receipt of a signal, the signal is not available, or it is not known whether the job terminated due to the receipt of a signal. In both cases *drmaa_wtermsig*(3) will not provide signal information. A non-zero value returned in *signaled* indicates signal information can be retrieved by means of *drmaa_wtermsig*(3).

drmaa_wcoredump()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wcoredump() function evaluates into the integer pointed to by *core_dumped* a non-zero value if a core image of the terminated job was created.

drmaa_wexitstatus()

If *drmaa_wifexited*(3) returned a non-zero value in the *exited* parameter, the drmaa_wexitstatus() function evaluates into the integer pointed to by *exit_code* the exit code that the job passed to *exit*(2) or the value that the child process returned from main.

drmaa_wtermsig()

If *drmaa_wifsignaled*(3) returned a non-zero value in the *signaled* parameter, the drmaa_wtermsig() function evaluates into *signal* up to *signal_len* characters of a string representation of the signal that caused the termination of the job. For signals declared by POSIX.1, the symbolic names are returned (e.g., SIGABRT, SIGALRM). For signals not declared by POSIX, any other string may be returned.

ENVIRONMENTAL VARIABLES

- *SGE_ROOT* Specifies the location of the Altair Grid Engine standard configuration files.
- *SGE_CELL* If set, specifies the default Altair Grid Engine cell to be used. To address a Altair Grid Engine cell Altair Grid Engine uses (in the order of precedence):

  ```
  The name of the cell specified in the environment variable SGE_CELL, if it is set.

  The name of the default cell, i.e. *default*.
  ```

- *SGE_DEBUG_LEVEL* If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
- *SGE_QMASTER_PORT* If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

RETURN VALUES

Upon successful completion, drmaa_run_job(), drmaa_run_bulk_jobs(), and drmaa_get_next_job_id() return DRMAA_ERRNO_SUCCESS. Other values indicate an error. Up to *error_diag_len* characters of error related diagnosis information is then provided in the buffer *error_diagnosis*.

ERRORS

The drmaa_synchronize(), drmaa_wait(), drmaa_wifexited(), drmaa_wexitstatus(), drmaa_wifsignaled(), drmaa_wtermsig(), drmaa_wcoredump(), and drmaa_wifaborted() will fail if:

DRMAA_ERRNO_INTERNAL_ERROR

Unexpected or internal DRMAA error, like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE

The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT

The input value for an argument is invalid.

DRMAA_ERRNO_NO_ACTIVE_SESSION

Failed because there is no active session.

DRMAA_ERRNO_NO_MEMORY

Failed allocating memory.

The drmaa_synchronize() and drmaa_wait() functions will fail if:

DRMAA_ERRNO_EXIT_TIMEOUT

Time-out condition.

DRMAA_ERRNO_INVALID_JOB

The job specified by the does not exist.

The drmaa_wait() will fail if:

DRMAA_ERRNO_NO_RUSAGE

This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

SEE ALSO

drmaa_submit(3)

COPYRIGHT

sge_jsv_script_interface

NAME

Altair Grid Engine Job Submission Verifier Scripting Interface

SYNOPSYS

Main Loop and Callback Functions

jsv_main();
jsv_on_start();
jsv_on_verify();

Accessing Non-List Based Parameters and Values

jsv_is_param(param_name);
jsv_get_param(param_name);
jsv_set_param(param_name, param_value);
jsv_del_param(param_name);

Accessing List Based Parameters and Values

jsv_sub_is_param(param_name, variable_name);
jsv_sub_get_param(param_name, variable_name);
jsv_sub_add_param(param_name, variable_name, variable_value);
jsv_sub_del_param(param_name, variable_name);

Accessing Job Environment

jsv_is_env(variable_name);
jsv_get_env(variable_name);
jsv_add_env(variable_name, variable_value);
jsv_mod_env(variable_name, variable_value);
jsv_del_env(variable_name);

Reporting Results

jsv_accept(message);

jsv_correct(message);

jsv_reject(message);

jsv_reject_wait(message);

Logging Functions

jsv_log_info(message);

jsv_log_warning(message);

jsv_log_error(message);

Protocol Functions

jsv_send_env();

jsv_set_timeout(new_timeout);

Others

jsv_clear_params();

jsv_clear_envs();

jsv_show_params();

jsv_show_envs();

DESCRIPTION

The functions documented here implement the server side of the JSV protocol as it is described in the man page sge_jsv(1). These script functions are available in Bourne shell, python2.7 or Perl scripts after sourcing/including the files jsv_include.sh, JSV.py or JSV.pm. The files and corresponding JSV script templates are located in the directory $SGE_ROOT/util/resources/jsv.

jsv_clear_params()

This function clears all received job parameters that were stored during the last job verification process.

jsv_clear_envs()

This function clears all received job environment variables that were stored during the last job verification process.

jsv_show_params()

A call of this function reports all known job parameters to the counterpart of this script (client or master daemon thread). This parameters will be reported as info messages and appear either in the stdout stream of the client or in the message file of the master process.

jsv_show_envs()

A call of this function reports all known job environment variables to the counterpart of this script (client or master daemon thread). They will be reported as info messages and appear in the stdout stream of the client or in the message file of the master process.

jsv_is_param(*param_name*)

This function returns whether or not a job parameter named *param_name* is available for the job which is currently being verified. Either the string *true* or *false* will be returned. The availability/absence of a job parameter does not mean that the corresponding command line switch was used/not used.

Allowed parameters names are the names of qsub(1) switches or similar names documented in the corresponding section of the qsub(1) man page. There are also pseudo parameters available that are described in sge_jsv(5).

Please note that the list of parameters below is not necessarily complete. Please refer to the corresponding man pages.

|————————|————————————————————| | *param_name* | command line switch/description | |————————|————————————————————| | a | | | ac | combination of **-ac**, **-sc**, **-dc** | | ar | | | A | | | b | | | c | | | ckpt | | | cwd | | | display | | | dl | | | e | | | h | | | hold_jid | | | hold_jid_ad | | | i | | | l_hard | **-l** or **-hard** followed by **-l** | | l_soft | **-soft** followed by **-l** | | j | | | jc | | | js | | | m | | | M | | | masterl | | | masterq | | | N | | | notify | | | now | | | N | | | o | | | ot | | | P | | | pe_min_<N>| min value of N'th-PE-range | | pe_max_<N>| max value of N'th-PE-range | | pe_n | Number of ranges spec. with **-pe** | | pe_n | Number of ranges spec. with **-pe** | | pe_name | PE name specified with **-pe** | | q_hard | **-q** or **-hard** followed by **-q** | | q_soft | **-soft** followed by **-q** | | R | | | r | | | shell | | | S | | | t | | | w | | | wd | | | … | | | | | CLIENT | | | CONTEXT | | | GROUP | | | VERSION | | | JOB_ID | | | SCRIPT | | | CMDARGS | | | CMDARG<N> | where <N> is a positive number | | USER | | | SCRIPTSIZE | | | … | | | | |————————|————————————————————|

jsv_get_param(*param_name*)

This function returns the value of a specific job parameter (*param_name*).

This value is only available if the function **jsv_is_param()** returns *true*. Otherwise an empty string is returned.

Allowed parameters names and values can be found in qsub(1) and sge_jsv(5).

jsv_set_param(*param_name*, *param_value*)

This function changes the job parameter *param_name* to *param_value*.

For boolean parameters that only accept the values *yes* or *no* and for the parameters *c* and *m* it is not allowed to pass an empty string as *param_value*.

If *param_value* is an empty string then the corresponding job parameter will be deleted as if **jsv_del_param()** was called.

param_value will replace/set the corresponding value in the job template and if the job is accepted by the system later on then the job will behave as if the corresponding command line switch or switch combination was specified during job submission.

Allowed parameters names and values can be found in qsub(1) and sge_jsv(5).

jsv_del_param(*param_name*)

This function deletes the job parameter *param_name*.

Allowed parameters names can be found in qsub(1) and sge_jsv(5).

jsv_sub_is_param(*param_name*, *variable_name*)

Some job parameters are lists that can contain multiple variables with an optional value.

This function returns *true* if a job parameters list contains a *variable* and *false* otherwise. *false* might also indicate that the parameter list itself is not available. Use the function jsv_is_param() to check if the parameter list is not available.

The following parameters are list parameters. The second columns within following table describes corresponding variable names to be used when jsv_sub_is_param() is called. The third column contains a dash (-) if there is no value (variable_value) allowed when the functions jsv_sub_add_param() or it indicates that jsv_sub_get_param() will return always an empty string. A question mark (?) shows that the value is optional.

|————————|————————————|————————-| | *param_name* | *variable_name* | *variable_value* | |————————|————————————|————————-| | ac | job context variable name | | | hold_jid | job identifier | - | | l_hard | complex attribute name | ? | | l_soft | complex attribute name | ? | | M | mail address | - | | masterl | complex attribute name | ? | | masterq | cluster queue name or | - | | | queue instance name | | | q_hard | cluster queue name | - | | | queue instance name | | | q_soft | cluster queue name or | - | | | queue instance name | | |————————|————————————|————————-|

jsv_sub_get_param(*param_name*, *variable_name*)

Some job parameters are lists that can contain multiple variables with an optional value.

This function returns the value of such a variable (*variable_name*) for that parameter (*param_name*).
For sub list elements that have no value an empty string will be returned.

Find a list of allowed parameter names and variable names in the section for the function jsv_sub_is_param() or in the man pages qsub(1) or sge_jsv(5).

jsv_sub_add_param(*param_name*, *variable_name*, *variable_value*)

Some job parameters are lists that can contain multiple variables with an optional value.

This function either adds a new variable (*variable_name*) with a new value (*variable_value*) or it modifies the value if the variable exists already in the list parameter with the name *param_name*. *variable_value* is optional. In case it is omitted, the variable has no value.

Find a list of allowed parameter names (*param_name*) and variable names (*variable_name*) in the section for the function jsv_sub_is_param() or in the man pages qsub(1) and sge_jsv(1).

jsv_sub_del_param(*param_name*, *variable_name*)

Some job parameters are lists which can contain multiple variables with an optional value.

This function deletes a variable (*variable_name*) and if available the corresponding value. If the variable does not exist in the job parameter named *param_name* then the command will be ignored.

Find a list of allowed parameter names (*param_name*) and variable names (*variable_name*) in the section for the function jsv_sub_is_param() or in the man pages qsub(1) and sge_jsv(1).

jsv_is_env(*variable_name*)

In order to be able to see job environment information a JSV script has to call **jsv_send_env()** in the callback function **jsv_on_start().**

This function can be used to check if a job specification contains an environment variable named *variable_name* that will be exported to job environment when the job accepted and later on started.

If this function returns *true*, then the job environment variable with the name *variable_name* exists in the job currently being verified and **jsv_get_env()** can be used to retrieve the value of that variable.

If the function returns *false*, then the job environment variable does not exist or the job environment information has not been enabled by calling **jsv_send_env()** in **jsv_on_start()** in the JSV script.

jsv_get_env(*variable_name*)

In order to be able to see job environment information a JSV script has to call **jsv_send_env()** in the callback function **jsv_on_start().**

To check if a variable is set the function **jsv_is_env()** can be called.

This function returns the value of a job environment variable (*variable_name*) if it is defined.

To change the value of a variable use the function **jsv_mod_env()**, to add a new variable with value, call the function **jsv_add_env()**. To delete a environment variable, use the function **jsv_del_env()**.

jsv_add_env(*variable_name*, *variable_value*)

In order to be able to see job environment information a JSV script has to call **jsv_send_env()** in the callback function **jsv_on_start().**

If environment information has been enabled for the JSV then a call of this function adds a new variable named *variable_name* with the value *variable_value* to the set of environment variables that will be exported to the job when it is accepted by the system and later on started.

variable_value is optional. If there is an empty string passed then the variable is defined without value. Newlines '\n' within the *variable_value* will be considered.

If an environment variable with the name *variable_name* already exists then this function will behave as if the function **jsv_mod_env()** was called.

To check if a variable exists the function **jsv_is_env()** can be used. To change the value of a variable use the function **jsv_mod_env()**. To delete a environment variable, use the function **jsv_del_env()**.

jsv_mod_env(*variable_name*, *variable_value*)

In order to be able to see job environment information a JSV script has to call **jsv_send_env()** in the callback function **jsv_on_start().**

If environment information has been enabled for the JSV then a call of this function modifies and existing environment variable to the set of environment variables that will be exported to the job when it is accepted by the system and later on started.

variable_value is optional. If there is an empty string passed then the variable is defined without value. Newlines '\n' within the *variable_value* will be considered.

To check if a variable exists the function **jsv_is_env()** can be used. To add a new variable call the function **jsv_add_env()**. To delete a environment variable, use the function **jsv_del_env()**.

jsv_del_env(*variable_name*)

In order to be able to see job environment information a JSV script has to call **jsv_send_env()** in the callback function **jsv_on_start().**

If environment information has been enabled for the JSV then a call of this function removes a job environment variable (*variable_name*) from the set of environment variables that will be exported to the job when it is accepted by the system and later on started.

If *variable_name* does not already exist, then the command is ignored.

To check if a variable exists the function **jsv_is_env()** can be used. To change the value of a variable use the function **jsv_mod_env()**, to add a new variable with value, call the function **jsv_add_env()**. To delete a environment variable, use the function **jsv_del_env()**.

jsv_accept()

This function can only be used in **jsv_on_verify()**. After it has been called, the function **jsv_on_verify()** has to return immediately.

A call to this function indicates that the job that is currently being verified should be accepted as it was initially provided. All job modifications that might have been applied in **jsv_on_verify()** before this function was called, are then ignored.

Instead of calling **jsv_accept()** in **jsv_on_verify()** also the functions **jsv_correct()**, **jsv_reject()** or **jsv_reject_wait()** can be called, but only one of these functions can be used at a time.

jsv_correct()

This function can only be used in **jsv_on_verify()**. After it has been called, the function **jsv_on_verify()** has to return immediately.

A call to this function indicates that the job that is currently being verified has to be modified before it can be accepted. All job parameter modifications that were previously applied will be committed and the job will be accepted. Accept in that case means that the job will either be passed to the next JSV instance for modification or that it is passed to that component in the master daemon that adds it to the master data store when the last JSV instance has verified the job.

Instead of calling **jsv_correct()** in **jsv_on_verify()**, the functions **jsv_accept()**, **jsv_reject()** or **jsv_reject_wait()** can be called, but only one of these functions can be used.

jsv_reject(*message*)

This function can only be used in **jsv_on_verify()**. After it has been called the function **jsv_on_verify()** has to return immediately.

The job that is currently being verified will be rejected. *message* will be passed to the client application that tried to submit the job. Command line clients like **qsub(1)** will print that *message* to *stdout* to inform the user that the submission has failed.

jsv_reject_wait() should be called if the user may try to submit the job again. **jsv_reject_wait()** indicates that the verification process might be successful in the future.

Instead of calling **jsv_reject()** in **jsv_on_verify()** also the functions **jsv_accept()**, **jsv_correct()** or **jsv_reject_wait()** can be also called, but only one of these functions can be used.

jsv_reject_wait(*message*)

This function can only be used in **jsv_on_verify()**. After it has been called the function **jsv_on_verify()** has to return immediately.

The job which is currently verified will be rejected. *message* will be passed to the client application, that tries to submit the job. Command line clients like qsub will print that *message* to *stdout* to inform the user that the submission has failed.

This function should be called if the user who tries to submit the job might have a chance to submit the job later. **jsv_reject()** indicates that the verified job will also be rejected in future.

Instead of calling **jsv_reject_wait()** in **jsv_on_verify()** the functions **jsv_accept()**, **jsv_correct()** or **jsv_reject()** can be also called, but only one of these functions can be used.

jsv_log_info(*message*)

This function sends an info *message* to the client or master daemon instance that started the JSV script.

For client JSVs, this means that the command line client will get the information and print it to the *stdout* stream. Server JSVs will print that *message* as an *info* message to the master daemon message file.

If *message* is missing then and empty line will be printed.

jsv_log_warning(*message*)

This function sends a warning message to the client or master daemon instance that started the JSV script.

For client JSVs, this means that the command line client will get the information and print it to the *stdout* stream. Server JSVs will print that *message* as an *warning* message to the master daemon message file.

If *message* is missing then and empty line will be printed.

jsv_log_error(*message*)

This function sends an error message to the client or master daemon instance that started the JSV script.

For client JSVs, this means that the command line client will get the information and print it to the *stdout* stream. Server JSVs will print that *message* as an *error* message to the master daemon message file.

If *message* is missing then and empty line will be printed.

jsv_send_env()

This function can only be used in **jsv_on_start()**. If it is used there, then the job environment information will be available in **jsv_on_verify()** for the next job that is scheduled to be verified.

This function must be called for the functions **jsv_show_envs()**, **jsv_is_env()**, **jsv_get_env()**, **jsv_add_env()** and **jsv_mod_env()** to behave correctly.

Job environments might become very big (10K and more). This will slow down the executing component (submit client or master daemon thread). For this reason, job environment information is not passed to JSV scripts by default.

Please note also that the data in the job environment can't be verified by Altair Grid Engine and might therefore contain data which could be misinterpreted in the script environment and cause security issues or failure of the JSV script.

jsv_set_timeout(*timeout*)

This function can only be used in **jsv_on_start()**.

The first argument of the function has to specify a *timeout* value in seconds. The value has to be greater than *0*. This value will be used instead of the value *SGE_JSV_TIMEOUT* that might have been defined in the environment of clients that execute client JSVs.

timeout will overwrite the parameter *jsv_timeout* that might exist as *qmaster_param* of the global configuration that defines the default timeout for server JSV and it will also overwrite the builtin default of *10* seconds in client and server JSV that is used when *SGE_JSV_TIMEOUT* and *jsv_timeout* are not defined.

In case that the response time of the JSV is longer than the timeout value specified, this will cause the JSV to be aborted and restarted. If *timeout* has been reached, the JSV will only try to restart once for one job verification, if the *timeout* is reached again an error will occur.

jsv_main()

This function has to be called in a main function in JSV scripts. It implements the JSV protocol that is documented in sge_jsv(5) and performs the communication with client and server components which might start JSV scripts.

This function does not return immediately. It returns only when the **QUIT** command is send by the client or server component and that is exchanged as part of the communication protocol.

During the communication with client and server components, this function triggers two callback func- tions for each job that should be verified. First **jsv_on_start()** and later on **jsv_on_verify()**.

jsv_on_start()* can be used to initialize certain things that might be needed for the verification process. jsv_on_verify()** has to do the verification process itself.

The function **jsv_send_env()** can be called in **jsv_on_start()** so that the job environment will be available in **jsv_on_verify()** for jobs to be verifies. This environment information would otherwise not be available in JSV per default for performance reason.

The following function can only be used in **jsv_on_verify()**. Simple job parameters can be accessed/modified with: **jsv_is_param()**, **jsv_get_param()**, **jsv_set_param()** and **jsv_del_param()**.

List based job parameters can be accessed with: **jsv_sub_is_param()*, **jsv_sub_get_param()**, **jsv_sub_add_param()** and **jsv_sub_del_param()**.

If the environment was requested with **jsv_send_env()** in **jsv_on_start()** then the environment can be accessed/modified with the following commands: **jsv_is_env()*, **jsv_get_env()**, **jsv_add_env()**, **jsv_mod_env()** and **jsv_del_env()**.

Jobs can be accepted/rejected with the following: **jsv_accept()**, **jsv_correct()**, **jsv_reject()** and **jsv_reject_wait()**.

The following functions send messages to the calling component of a JSV that will either appear on the *stdout* stream of the client or in the master *message* file. This is especially useful when new JSV scripts should be tested: **jsv_show_params()**, **jsv_show_envs()**, **jsv_log_info()**, **jsv_log_warning()** and **jsv_log_error()**.

jsv_on_start()

This is a callback function that has to be defined by the creator of a JSV script. It is called for every job, short time before the verification process of a job starts.

Within this function **jsv_send_env()** can be called to request job environment information for the next job that is scheduled to be verified.

jsv_set_timeout() can also be used to define a new *timeout* value that should overwrite predefined timeout values in the system.

After this function returns **jsv_on_verify()** will be called.

jsv_on_verify()

This is a callback function that has to be defined by the creator of a JSV script and that will be called by **jsv_main()** automatically. It is called for every job and when it returns the job will either be accepted or rejected.

Find implementation examples in the directory $SGE_ROOT/util/resources/jsv.

The logic of this function completely depends on the creator of this function. The creator has only to take care that one of the functions **jsv_accept()**, **jsv_reject()**, **jsv_reject_wait()** or **jsv_correct()** is called before the function returns.

The allowed execution time for this function is limited by *timeout* values. Find more information about predefined timeout values in the section for **jsv_set_timeout()** that also allows to increase the timeout.

It is recommended to keep the execution time for **jsv_on_verify()** as small as possible. Not doing so will cause negative performance impacts not only for job submissions but also for the overall cluster performance.

Time intensive tasks that do preparational work for the JSV should be done before **jsv_main()** is called.

If nevertheless costly things need to be done in **jsv_on_verify()** then it is recommended to increase the number of worker threads in the bootstrap(5) file of Altair Grid Engine.

EXAMPLES

Accessing List Based Parameters

Find in the table below the returned values for the "*is*" and "*get*" functions when following job is submitted:

qsub -l mem=1G,mem2=200M ...

|———————————————-|———————–| | function call | returned value | |———————————————-|———————–| | jsv_is_param(l_hard) | "true" | | jsv_get_param(l_hard) | "mem=1G,mem2=200M" | | jsv_sub_is_param(l_hard,mem) | "true" | | jsv_sub_get_param(l_hard,mem) | "1G" | | jsv_sub_get_param(l_hard,mem3) | "false" | | jsv_sub_get_param(l_hard,mem3) | "" | |———————————————-|———————–|

More Examples

More examples can be found in scripts located in $SGE_ROOT/util/resources/jsv of the Altair Grid Engine installation.

COPYRIGHT

sge_access_list

NAME

sge_access_list(5) - Altair Grid Engine access list file format

DESCRIPTION

Access lists are used in Altair Grid Engine to define access permissions of users to the cluster via the global configuration (see *sge_conf*(5)) and job classes (see *sge_job_class*(5)). They define access permissions to hosts (see *sge_host_conf*(5)), queues (see *sge_queue_conf*(5)), parallel environments (see *sge_pe*(5)), projects (see *sge_project*(5)) and they allow to define limits for multiple users via resource quotas (see *sge_resource_quota*(5))

A list of currently configured access lists can be displayed via the *qconf*(1) **-sul** option. The contents of each enlisted access list can be shown via the **-su** switch. The output follows the *access_list* format description. New access lists can be created and existing can be modified via the **-au** and **-du** options to *qconf*(1).

Departments are a special form of access list that additionally allow assignment of functional shares and override tickets.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

There are several predefined access lists that gain access to certain functionalities of a Altair Grid Engine system:

arusers

The access control list allows users that are referenced to submit and control advance reservations (see *qrsub*(1)).

deadlineusers

Users mentioned in this access control list can specify a deadline for jobs either during the submission with *qsub*(1) **-dl** or after submission with the *qalter*(1) **-dl** switch.

sessionusers

GDI sessions have been introduced with Altair Grid Engine 8.2. Such objects can be created, modified and deleted by managers or users that are members of the **sessionusers** access

control list. The list will be defined during a default installation of Altair Grid Engine but it will be empty. Either users can be added to give them permissions to create, modify and deleted own session objects or the **sessionusers** access control list can be deleted to give all users permissions.

sudomasters

see **sudoers** below.

sudoers

sudomasters and **sudoers** access lists are used in combination with the Altair Grid Engine Rest Service. Users mentioned in **sudomasters** can trigger commands on behalf of different users that have to be specified in **sudoers** so that corresponding Altair Grid Engine requests are accepted and executed. This functionality is available since Altair Grid Engine 8.3.

FORMAT

The following list of *access_list* parameters specifies the *access_list* content:

name

The name of the access list as defined for *userset_name* in *sge_types*(1).

type

The type of the access list, currently one of *ACL,* or *DEPT* or a combination of both in a comma separated list. Depending on this parameter the access list can be used as access list only or as a department.

oticket

The amount of override tickets currently assigned to the department.

fshare

The current functional share of the department.

entries

The entries parameter contains the comma separated list of those UNIX users (see *user_name* in *sge_types*(1)) or those primary UNIX groups that are assigned to the access list or the department. By default only a user's primary UNIX group is used; secondary groups are ignored as long as the *qmaster_param* **ENABLE_SUP_GRP_EVAL** is not defined. Only symbolic names are allowed. A group is differentiated from a user name by prefixing the group name with a '@' sign. Pure access lists allow enlisting any user or group in any access list.

When using departments, each user or group enlisted may only be enlisted in one department, in order to ensure a unique assignment of jobs to departments. The algorithm first searches for the user name in ACLs of type DEPT and if the user is not found it searches for the primary group id to identify the department of the user's job. To jobs whose users do not match with any of the users or groups enlisted under entries the *defaultdepartment* is assigned, if existing. If no department is found and the "defaultdepartment" ACL does not exist job submission is rejected.

When ENABLE_SUP_GRP_EVAL=1 is configured only the primary group id of a user is used to assign the department of a user's job when groups are used.

SEE ALSO

sge_host_conf(5), *sge_conf*(5), *sge_intro*(1), *sge_job_class*(5), *sge_types*(1), *sge_project*(5), *qconf*(1), *sge_pe*(5), *sge_queue_conf*(5).

COPYRIGHT

sge_accounting

NAME

sge_accounting(5) - Altair Grid Engine accounting file format

DESCRIPTION

An accounting record is written to the Altair Grid Engine accounting file for each job having finished. The accounting file is processed by qacct(1) to derive accounting statistics.

FORMAT

Each job is represented by a line in the accounting file. Empty lines and lines which contain one character or less are ignored. Accounting record entries are separated by colon (':') signs.

The entries listed here are not necessarily shown in the same order in the qacct(1) output. The numbers behind field names showing the position of the value in an accounting record.

qname (1)

Name of the cluster queue in which the job has run.

hostname (2)

Name of the execution host.

group (3)

The effective group id of the job owner when executing the job.

owner (4)

Owner of the Altair Grid Engine job.

job_name (5)

Job name.

job_number (6)

Job identifier - job number.

account (7)

An account string as specified by the qsub(1) or qalter(1) *-A* option.

priority (8)

Priority value assigned to the job corresponding to the priority parameter in the queue configuration (see *sge_queue_conf*(5)).

submission_time (9)

Submission time (64bit GMT unix time stamp in milliseconds).

start_time (10)

Start time (64bit GMT unix time stamp in milliseconds).

end_time (11)

End time (64bit GMT unix time stamp in milliseconds).

failed (12)

A non-0 failed status indicates a problem with job execution. See sge_diagnostics(5) for a description of the failed values. Some failed values indicate that the usage value and exit status of a job are not valid.

exit_status (13)

Exit status of the command or Altair Grid Engine specific status in case of certain error conditions. An exit status > 128 means that the job was killed by a signal. The exit status is 128 + signal number. Intermediate accounting records for jobs running around midnight have an exit status of -1.

For example: If a job dies through signal 9 (SIGKILL) then the exit status becomes 128 + 9 = 137.

ru_wallclock (14)

Difference between *end_time* and *start_time* (see above). This value is measured by sge_shepherd daemon

The remainder of the accounting entries follows the contents of the standard UNIX rusage structure as described in getrusage(2). Depending on the operating system where the job was executed some of the fields may be 0. The following entries are provided:

- ru_utime (15)
- ru_stime (16)
- ru_maxrss(17)
- ru_ixrss (18)
- ru_ismrss(19)
- ru_idrss (20)
- ru_isrss (21)
- ru_minflt(22)
- ru_majflt(23)
- ru_nswap (24)
- ru_inblock(25)
- ru_oublock(26)
- ru_msgsnd (27)
- ru_msgrcv (28)
- ru_nsignals(29)
- ru_nvcsw (30)
- ru_nivcsw (31)

On Windows, only the values *ru_wallclock*, *ru_utime* and *ru_stime* are accounted. These values are the final usage values of the Windows Job object that is used to reflect the Altair Grid Engine Job, not the sum of the usage of all processes.

project (32)

The project which was assigned to the job.

department (33)

The department which was assigned to the job.

granted_pe (34)

The parallel environment which was selected for that job.

slots (35)

The number of slots which were dispatched to the job by the scheduler.

task_number (36)

Array job task index number.

cpu (37)

The cpu time usage in seconds.

mem (38)

The integral memory usage in Gbytes cpu seconds.

io (39)

The amount of data transferred in Gbytes. On Linux data transferred means all bytes read and written by the job through the *read()*, *pread()*, *write()* and *pwrite()* systems calls.

On Windows this is the sum of all bytes transferred by the job by doing write, read and other operations. It is not documented what these other operations are.

category (40)

A string specifying the job category.

iow (41)

The io wait time in seconds.

pe_taskid (42)

If this identifier is set the task was part of a parallel job and was passed to Altair Grid Engine Engine via the “qrsh -inherit” interface.

maxvmem (43)

The maximum vmem size in bytes.

arid (44)

Advance reservation identifier. If the job used resources of an advance reservation then this field contains a positive integer identifier otherwise the value is *0*.

ar_submission_time (45)

If the job used resources of an advance reservation then this field contains the submission time (64bit GMT unix time stamp in milliseconds) of the advance reservation, otherwise the value is *0* .

job_class (46)

If the job has been running in a job class, the name of the job class, otherwise *NONE* .

qdel_info (47)

If the job (the array task) has been deleted via qdel, `<username>@<hostname>`, else *NONE*. If qdel was called multiple times, every invocation is recorded in a comma separated list.

maxrss (48)

The maximum resident set size in bytes.

maxpss (49)

The maximum proportional set size in bytes.

submit_host (50)

The submit host name.

cwd (51)

The working directory the job ran in as specified with qsub(1) / qalter(1) switches *-cwd* and *-wd*. As the delimiter used by the accounting file (colon "**:**") can be part of the working directory all colons in the working directory are replaced by ASCII code 255.

submit_cmd (52)

The submit command (submit_cmd) represents the command which was used for job submission. It does not include changes which have been done by the sge_request files or JSV scripts.

wallclock (53)

The wallclock time the job spent in running state. Time spent in prolog, epilog, pe_start and pe_stop scripts also counts as wallclock time. Times during which the job was suspended are not counted as wallclock time. This value is measured by execution daemon.

ioops (54)

The number of io operations.

bound_cores (55)

The socket and core number which was bound by this job (depends on ENABLE_BINDING_RECORDS qmaster_params setting).

resource_map (56)

The assigned Resource Map ids (depends on ENABLE_BINDING_RECORDS qmaster_params setting).

devices (57)

The assigned devices (from Resource Map **device** parameter, depends on ENABLE_BINDING_RECORDS qmaster_params setting).

gpus (58)

The assigned GPUs (from Resource Map **amd_id**, **cuda_id** or **xpu_id** parameter).

gpu_usage (59)

The GPU specific usage values that were reported for the job.

cgroups_failcnt (60)

Number of memory limit hits. If cgroups version 1 is used, this value is read from the file memory.failcnt. If cgroups version 2 is used, this value is read from the memory.events (property “max”).

cgroups_failcnt_swp (61)

Number of memory and swap limit hits. If cgroups version 1 is used, this value is read from the file memory.memsw.failcnt. If cgroups version 2 is used, this value is read from the file memory.swap.events (property “max”).

cgroups_max_mem (62)

Maximum recorded memory usage (cgroups). If cgroups version 1 is used, this value is read from the file memory.max_usage_in_bytes. If cgroups version 2 is used, this value is read from the file memory.peak.

Note: memory.peak is not available on all cgroups v2 systems. If it is not available, Altair Grid Engine will use the value from memory.current to calculate the maximum memory usage.

cgroups_max_mem_swp (63)

Maximum recorded memory and swap usage (cgroups). If cgroups version 1 is used, this value is read from the file memory.memsw.max_usage_in_bytes. If cgroups version 2 is used, this value is calculated as memory.swap.peak + memory.peak.

Note: memory.swap.peak and memory.peak are not available on all cgroups v2 systems. If the files are not available, Altair Grid Engine will use the values from memory.current and memory.swap.current to calculate the maximum memory usage.

cgroups_max_proc_mem (64)

Maximum of the cgroups_memory load value.

If cgroups version 1 is used, cgroups_memory is calculated from values in the cgroups file memory.stat as cgroups_memory = total_active_anon + total_inactive_anon + total_unevictable

If cgroups version 2 is used, cgroups_memory is calculated from values in the cgroups file memory.stat as cgroups_memory = active_anon + inactive_anon + unevictable

hold_jid (65)

The job ids which this job holds for.

orig_exec_time (66)

The initially requested execution time when the job was submitted (from **-a date_time** submission option).

exec_time (67)

The requested execution time (from **-a date_time** submission option).

ar_name (68)

The name of the Advance Reservation of the job.

hard_resources (69)

The hard requests of the job (from **[-hard] -l** submission option).

soft_resources (70)

The soft requests (from **-soft -l** submission option).

hard_queues (71)

The hard requested queues (from **[-hard] -q** submission option).

soft_queues (72)

The soft requested queues (from **-soft -q** submission option).

granted_req. (73)

The granted requests (may be different to the initial requests of the job, depends on ENABLE_BINDING_RECORDS qmaster_params setting). Each "attribute=value" pair is prefixed by "array task ID,parallel task ID:".

exec_host_list (74)

List of execution hosts the job was running on.

cgroups_cpu_usage (75)

Cpu accounting value reported by cgroups. This value is only reported if the cgroups parameter **cpuacct** is set to **true**. If cgroups version 1 is used, this value is read from the file cpuacct.usage. If cgroups version 2 is used, this value is read from the file cpu.stat (property "usage_usec").

COPYRIGHT

sge_aliases

NAME

sge_aliases - Altair Grid Engine path aliases file format

DESCRIPTION

The Altair Grid Engine path aliasing facility provides administrators and users with the means to reflect complicated and in-homogeneous file system structures in distributed environments (such as user home directories mounted under different paths on different hosts) and to ensure that Altair Grid Engine is able to locate the appropriate working directories for executing batch jobs.

There is a system global path aliasing file and a user local file. sge_aliases defines the format of both:

- Blank lines and lines with a '#' sign in the first column are skipped.
- Each line other than a blank line or a line lead by '#' has to contain four strings separated by any number of blanks or tabs.
- The first string specifies a source-path, the second a submit-host, the third an execution-host and the fourth the source-path replacement.
- Both the submit- and the execution-host entries may consist of only a '*' sign which matches any host.

If the -cwd flag (and only if - otherwise the user's home directory on the execution host is selected to execute the job) to qsub(1) was specified, the path aliasing mechanism is activated and the files are processed as follows:

- After qsub(1) has retrieved the physical current working directory path, the cluster global path aliasing file is read if present. The user path aliases file is read afterwards as if it were appended to the global file.
- Lines not to be skipped are read from the top of the file one by one while the translations specified by those lines are stored if necessary.
- A translation is stored only if the submit-host entry matches the host qsub(1) is executed on and if the source-path forms the initial part either of the current working directory or of the source-path replacements already stored.
- As soon as both files are read the stored path aliasing information is passed along with the submitted job.
- On the execution host, the aliasing information will be evaluated. The leading part of the current working directory will be replaced by the source-path replacement if the execution-host entry of the path alias matches the executing host. **Note:** The current working directory string will be changed in this case and subsequent path aliases must match the replaced working directory path to be applied.

EXAMPLES

The following is a simple example of a path aliasing file resolving problems with inhomogeneous paths if automount(8) is used:

```
==================================================
# Path Aliasing File
# src-path sub-host exec-host replacement
/tmp_mnt/ * * /
# replaces any occurrence of /tmp_mnt/ by /
# if submitting or executing on any host.
# Thus paths on nfs server and clients are the same
==================================================
```

FILES

`<sge_root>/<cell>/common/sge_aliases` - global aliases file

$HOME/.sge_aliases - user local aliases file

SEE ALSO

sge_intro(1), qsub(1), Altair Grid Engine Installation and Administration Guide

COPYRIGHT

sge_amd(5)

NAME

sge_amd - Management of AMD GPUs in Altair Grid Engine

DESCRIPTION

The sections below describe the setup and management of AMD GPUs in Altair Grid Engine.

OVERVIEW

Altair Grid Engine uses Resource Maps (complex with type `RSMAP`) to map physical GPUs to internal resources that can be used during the scheduling of jobs.

For an overview of Resource Maps and how to configure and request complexes see sge_resource_map(5) and sge_complex(5).

AMD ROCm/AMD-SMI

Altair Grid Engine uses ROCm or AMD-SMI to get information about installed AMD GPUs.

If installed, Altair Grid Engine will automatically load the ROCm or AMD-SMI library and set the load value m_gpu (i.e. the number of installed physical GPUs) and other GPU specific load values that can be displayed via `qconf -se <hostname>`.

The library also used to collect GPU specific usage values of jobs.

If both ROCm and AMD-SMI are installed on a host, Altair Grid Engine will prioritize loading the AMD-SMI library and will use ROCm only if loading AMD-SMI fails.

For a list of supported ROCm/AMD-SMI versions see the Release Notes.

Job Usage Statistics: ROCm/AMD-SMI can be used to collect GPU usage statistics of jobs. For more information see USAGE STATISTICS.

GPU Load Values

If ROCm/AMD-SMI is available, the following GPU specific load values will be reported for hosts with supported AMD GPUs. The load values are displayed in `qconf -se <hostname>`.

Value	Description
amd.devices	Number of installed AMD GPUs
amd.verstr	Installed driver version
amd.<gpu>.affinity	Affinity between GPU <gpu> and the installed CPU(s) as reported by `rocm-smi --showtopo` and `amd-smi static`
amd.<gpu>.gpu_temp	GPU temperature
amd.<gpu>.health	Health status of the GPU. Only reported when != 0
amd.<gpu>.mem_total	Total memory of the GPU
amd.<gpu>.mem_free	Free memory of the GPU
amd.<gpu>.mem_used	Used memory of the GPU
amd.<gpu>.name	Name of the GPUs
amd.<gpu>.path	Device path of the MIG (needed for cgroups)
amd.<gpu>.uuid	UUID of the GPU

SETUP

Resource Maps

For an overview of Resource Maps and how to configure complexes with type `RSMAP` see sge_resource_map(5).

The following additional parameter can be used to map GPUs to Resource Map ids:

- `amd_id`: Id of the GPU to which this id should be mapped (according to the output of `rocm-smi` or `amd-smi monitor`). The id will be used to export `ROCR_VISIBLE_DEVICES`.

Cgroups

Access to GPUs can be managed via cgroups to ensure that jobs are only able to access devices that were assigned to them.

If the cgroups_param **devices** is set to a list of device paths (separated by `|`), the cgroups devices subsystem/controller will be used to block access to GPUs in the list and jobs will only be able to access GPUs that were assigned to them via RSMAPs (see RMSAP parameter `device`)

The following configuration will block access to all AMD GPUs:

```
devices=/dev/dri/card[0-256]|/dev/dri/renderD*
```

If a RSMAP with the device `/dev/dri/card1|/dev/dri/renderD128` is assigned to a job, the job will be able to access the GPU to which this device path belongs. The device paths for each GPU are shown in the `qconf -se <hostname>` output.

Jobs that do not request RSMAPs will not be able to access any of the installed GPUs.

For instructions on how to enable cgroups and how to set the cgroup_param **devices**, see sge_conf(5).

SCHEDULING

Requesting GPUs

For a overview of Resource Maps and how to request complexes with type `RSMAP` see sge_resource_map(5).

The following additional parameters can be requested for GPUs and will have an effect on the scheduling result:

- `affinity`: see GPU-CPU Affinity

GPU-CPU Affinity

If a job requests a GPU and **affinity**, it will automatically be bound to the CPU cores that have a good affinity to the assigned GPU (see `rocm-smi --showtopo` or `amd-smi static`). This ensures that the data between the CPU and GPU is transferred in the fastest way possible.

The parameter supports two values:

- **1/true**: Affinity is requested as hard request
- **2**: Affinity is requested as a soft request

If affinity is requested as hard request, the job will not be scheduled unless there is a GPU/CPU pair with good affinity and enough free CPU cores available. If affinity is requested as a soft request, Altair Grid Engine will try to schedule the job with a GPU/CPU pair with good affinity, but schedule the job anyway without binding to any CPU cores, if not enough CPU cores are available and free. If less cores are needed the request can be combined with the **-binding** switch.

ROCR_VISIBLE_DEVICES

Altair Grid Engine can automatically export the environment variable `ROCR_VISIBLE_DEVICES` for jobs that request GPUs. For more information see sge_conf(5), `SET_ROCR_VISIBLE_DEVICES`.

USAGE STATISTICS

If enabled, ROCm/AMD-SMI is used to collect GPU specific usage values that are displayed in the gpu_job_usage section of qstat -j <job_id> and qacct -j <job_id>.

GPU Job Usage

The following usage values are available. The values are displayed in qstat -j <job_id> in the format <hostname>(amd.<usage_name>=<usage_value>).

Usage value	Description
sdma	SDMA usage in seconds
vram	VRAM usage
cu_occupancy	Compute Unit usage in percent

Which values to report is defined via the global/local configuration attribute *gpu_job_usage*. The following values can be configured, the default value is *none*.

Setting	Description
none	No gpu usage values will be reported
all	All implemented usage values will be reported, see the table above
<variable list>	Comma or space separated list of variable names, without *amd.* e.g. *sdma*, *vram*, ...

GPU Job Accounting

GPU usage values can also be written to the accounting file and displayed in qacct -j <job_id>.

Which values to write to the accounting file is defined via the global/local configuration attribute *gpu_job_accounting*.

The default value for **gpu_job_accounting** is *none*. All settings that are valid for **gpu_job_usage** are also valid for **gpu_job_accounting**.

NOTE: **gpu_job_usage** and **gpu_job_accounting** can be configured individually. If **gpu_job_accounting** is set to "none", no gpu specific usage values will be written to the accounting file, regardless of the configuration of **gpu_job_usage**.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), qsub(1), complex(5), host(5), sge_job_class(5), sge_resource_map(5).

COPYRIGHT

sge_bootstrap

NAME

bootstrap - Altair Grid Engine bootstrap file

DESCRIPTION

bootstrap contains parameters that are needed for the startup of Altair Grid Engine components. It is created during the sge_qmaster installation. Modifying *bootstrap* in a running system is not supported.

FORMAT

The paragraphs that follow provide brief descriptions of the individual parameters that compose the bootstrap configuration for a Altair Grid Engine cluster:

admin_user

Administrative user account used by Altair Grid Engine for all internal file handling (status spooling, message logging, etc.). Can be used in cases where the root account does not have the corresponding file access permissions (e.g. on a shared file system without global root read/write access).

Being a parameter set at installation time changing **admin_user** in a running system is not supported. Changing it manually on a shut-down cluster is possible, but if access to the Altair Grid Engine spooling area is interrupted, this will result in unpredictable behavior.

The **admin_user** parameter has no default value, but instead it is defined during the master installation procedure.

default_domain

Only needed if your Altair Grid Engine cluster covers hosts belonging to more than a single DNS domain. In this case it can be used if your hostname resolving yields both qualified and unqualified hostnames for the hosts in one of the DNS domains. The value of **default_domain** is appended to the unqualified hostname to define a fully qualified hostname. The **default_domain** parameter will have no effect if **ignore_fqdn** is set to “true”.

Being a parameter set at installation time changing **default_domain** in a running system is not supported. The default for **default_domain** is “none”, in which case it will not be used.

ignore_fqdn

Ignore fully qualified domain name component of hostnames. Should be set if all hosts belonging to a Altair Grid Engine cluster are part of a single DNS domain. It is switched on if set to either “true” or “1”. Switching it on may solve problems with load reports due to different hostname resolutions across the cluster.

Being a parameter set at installation time changing **ignore_fqdn** in a running system is not supported. The default for **ignore_fqdn** is “true”.

spooling_method

Defines how sge_qmaster(8) writes its configuration and the status information of a running cluster.

The available spooling methods are *lmdb*, *postgres* and *classic*.

spooling_lib

The name of a shared library containing the **spooling_method** to be loaded at sge_qmaster(8) initialization time. The extension characterizing a shared library (.so, .sl, .dylib etc.) is not contained in **spooling_lib**.

If **spooling_method** was set to *lmdb* during installation, **spooling_lib** is set to *libspoolb*, if *postgres* was chosen as **spooling_method**, **spooling_lib** is set to *libspoolp*, if *classic* was chosen as **spooling_method**, **spooling_lib** is set to *libspoolc*.

Not all operating systems allow the dynamic loading of libraries. On these platforms a certain spooling method (default: lmdb) is compiled into the binaries and the parameter **spooling_lib** will be ignored.

spooling_params

Defines parameters to the chosen spooling method.

Parameters that are needed to initialize the spooling framework, e.g. to open database files or to connect to a certain database server.

The spooling parameters value for spooling method *lmdb* is the path to the database directory, e.g. /sge_local/default/spool/qmaster/spooldb for spooling to a local filesystem.

Additional options can be configured after the path to the database directory separated by semicolon, e.g. /sge_local/default/spool/qmaster/spooldb;ASYNC=TRUE;FLUSH_CONFIG=TRUE.

Use lmdb spooling options with care, best contact Altair Grid Engine support for advice. The following options can be configured:

Option	Description
ASYNC	TRUE/FALSE: do asynchronous writes. This is significantly faster than synchronous writes. In case of a crash it can lead

Option	Description
	to a loss of the most recent spooling operations, though. Default: FALSE.
COMPRESSION	Compression level to be used by zlib compression of objects before writing them to the database. Valid values are [0..9]. 0 means no compression, level 1 is little but fastest compression, 9 gives the highest compression but is slowest. Default is 0, without compression.
FLUSH_CONFIG	TRUE/FALSE: in case of ASYNC=TRUE: Do an immediate sync after writing configuration objects. Default: TRUE.
FLUSH_INTERVAL	number of seconds: in case of ASYNC=TRUE: Sync in the given interval. Default: 2 seconds.
FLUSH_OPERATIONS	number of database operations done: In case of ASYNC=TRUE: sync after the given number of database operations. Default: 2000.
MAP_SIZE	number of pages: Initial size of the memory map. It is re-sized on demand. Default: Number of pages resulting in a map size of 2GB (depending on OS page size).
CHECK_INTERVAL	number of seconds: Interval for recurring database checks (check if size of memory map is still sufficient, check for stale reader processes). Default: 10 seconds.

For spooling method *postgres* the spooling parameters contain the PostgreSQL connection string in Keyword/Value format documented at http://www.postgresql.org/docs/9.3/interactive/libpq-connect.html#LIBPQ-CONNSTRING.

One spooling option can be added separated by semicolon:

Option	Description
COMPRESSION	Compression level to be used by zlib compression of objects before writing them to the database. Valid values are [0..9]. 0 means no compression, level 1 is little but fastest compression, 9 gives the highest compression but is slowest. Default is 0, without compression.

For spooling method *classic* the spooling parameters take the form `<common_dir>;<qmaster spool dir>`, e.g. /sge/default/common;/sge/default/spool/qmaster

One spooling option can be added separated by semicolon:

Option	Description
COMPRESSION	Compression level to be used by zlib compression of job objects and job scripts before writing them to disk as data files. Valid values are [0..9]. 0 means no compression, level 1 is little but fastest compression, 9 gives the highest compression but is slowest. Default is 0, without compression.

binary_path

The directory path where the Altair Grid Engine binaries reside. It is used within Altair Grid Engine components to locate and startup other Altair Grid Engine programs.

The path name given here is searched for binaries as well as any directory below with a directory name equal to the current operating system architecture. Therefore, /usr/sge/bin will work for all architectures, if the corresponding binaries are located in subdirectories named aix43, lx-amd64, lx-x86, hp11, hp11-64, sol-amd64, sol-sparc etc.

The default location for the binary path is `<sge_root>/bin`

qmaster_spool_dir

The location where the master spool directory resides. Only the sge_qmaster(8) and sge_shadowd(8) need to have access to this directory. The master spool directory - in particular the job_scripts directory and the messages log file - may become quite large depending on the size of the cluster and the number of jobs. Be sure to allocate enough disk space and regularly clean off the log files, e.g. via a *cron*(8) job.

Being a parameter set at installation time changing **qmaster_spool_dir** in a running system is not supported.

The default location for the master spool directory is `<sge_root>/<cell>/spool/qmaster`.

security_mode

The security mode defines the set of security features the installed cluster is using.

Possible security mode settings are none, afs, dce, kerberos, csp. (no additional security, AFS, DCE, KERBEROS, CSP security model). The only active supported security_mode is CSP. AFS, DCE, KERBEROS security is NOT supported anymore and might be removed in future releases.

communication_params

This setup parameter is used for setting up default communication parameters. The value specified after the keyword “communication params” has following syntax:

`<msg_mode=msg_mode_name[+msg_mode_name[+...]][;wp_threads=[<nr>|<min>-<max>]][;host_2_ip4_map_for`

msg_mode: List of supported message packing modes

msg_mode_name : `algorithm(min_len:<len_value>,level:<level_value>)|none`

wp_threads: Nr of fix enabled commlib work pool threads or range specification

host_2_ip4_map_format: Specification of string formats used for parsing IPv4 address out of hostnames

ip4_map_str: Specifies the position of the IPv4 numbers inside a typical hostname

If **communication params** is set to “none”, not specified or the line is missing completely the default “msg_mode=none” is used. In this case no special message packing mode is used.

The “msg_mode” parameter is used to specify the order of supported message modes. Message modes are used to define how messages that are sent or received via commlib.

With Altair Grid Engine 8.6.0 the only supported message mode algorithm is “zlib” for following architectures: darwin-x64, lx-amd64, lx-arm64, lx-arm7, lx-x86, sol-amd64, sol-sparc64, sol-x86.

Changes in the bootstrap file require a restart of all Altair Grid Engine daemons that are affected by the change.

The “zlib” algorithm is a compression method that will reduce the data size transferred via network commands.

Each message mode must have configured the following parameters:

Parameter	Description
algorithm	The message packing method that should be used (e.g. “zlib”).
min_len	The minimum message length that must be reached to use the specified algorithm. It is not supported to set the minimum length lower than 64 byte. The value is specified in bytes (It is supported to use the known Altair Grid Engine data unit letters like “K,k,M,m,..”).
level	This parameter is used for defining the compression level for compressing algorithms. The value “0” will select the default compression level supported by the specified algorithm. The values “1-9” will define how compression itself will perform. “1” would mean “best performance” and “9” would force the compression algorithm called to perform “best compression”.

Examples:

msg_mode	Description
none	Turn off compression at all times.
none+zlib(min_len:1K,level:0)	Turn off compression per default, but allow zlib compression for messages larger than 1024 bytes with default compression level if client requests it.
zlib(min_len:1K,level:0)+none	Turn on compression but also support no compression if a client does not support “zlib” algorithm.
zlib(min_len:1K,level:0)	Force compression, clients that do not support zlib cannot connect.

The optional “wp_threads” parameter is used to specify the nr of threads started for the commlib work pool. Since the bootstrap file is used by every component the default value is not to use any work pool. It is not supported to use more than 32 threads for the work

pool.

This parameter can be overwritten for sge_qmaster and sge_execd (See *sge_conf*(5) man page, qmaster_params/execd_params setting).

Examples:

wp_threads	Description
`0`	Using of additional commlib work pool disabled.
`4`	If commlib is initialized to support threads the commlib will create 4 additional threads for handling connections.
`0-4`	If the commlib is initialized to support threads it will automatically decide if additional work pool threads are needed or not. Commlib will not use more than 4 threads for the work pool.

The optional "host_2_ip4_map_format" parameter is used to specify strings used for parsing out the IPv4 address out of a hostname. If a hostname string contains the IPv4 address where the host can be reached, the hostname resolving call for this host is not necessary. It is possible to specify a list of mapping strings if they are separated by a ",".

Examples:

host_2_ip4_map_format	Description
`ip%d0-%d1-%d2-%d3-%d4`	The %d1-%d4 escape sequences are used to define the position of the IPv4 decimal address tuple. The %d0 sequence can be used to ignore a decimal digit. A host named "ip-192-168-11-1" would match this mapping format and the resulting IPv4 address used is 192.168.11.1.
`ip-%d1-%d2-%d3-%d4*`	The * character can be used to skip parsing of any additional characters of the hostname.

The environment variable **SGE_COMMLIB_COMMUNICATION_PARAMS** can be used to overwrite the "communication_params" line defined in the bootstrap configuration. This is useful e.g. to force a client to use "zlib" algorithm for msg_mode.

Additional information can be found in the *sge_diagnostics*(5) and *qping*(1) man page.

debug_params

This optional parameter can be set to enable special debugging options. The value specified after the keyword "debug_params" supports following settings:

`<dlock=qmaster_full|qmaster_default>`

dlock: Enable deadlock detection module:

Enabling the dlock feature might reduce the performance of the qmaster daemon.

dlock value	Description
qmaster_full	All qmaster threads that are monitored will also be registered at deadlock detection module
qmaster_default	Same setting as “qmaster_full” but without commlib read, write and workpool threads.

Additional information can be found in the *sge_diagnostics*(5) man page.

listener_threads

The default for this parameter will be set to 2 during installation process.

Listener threads are threads that do preliminary work for worker and reader threads running in sge_qmaster(8). At least two of them are required in a system.

pworker_threads

The default for this parameter will be set to 2 during installation process. The recommended value for this parameter is 2.

Defines the number of priority worker threads that are started during sge_qmaster(8) start. It is not possible to disable the priority worker thread pool and the number of threads part of that pool cannot be adjusted dynamically. The change of this parameter requires a restart of sge_qmaster(8).

All threads of the priority worker thread pool are responsible to handle priority requests that are triggered by commands in combination with the -preferred command line switch or by qmaster internal priority requests.

worker_threads

The default for this parameter will be set to 4 during installation process.

Defines the number of worker threads that are started during sge_qmaster(8) start. It is not possible to disable the worker thread pool and the number of threads part of that pool cannot be adjusted dynamically. The change of this parameter requires a restart of sge_qmaster(8).

The recommended value for this parameter is 4. If no server JSV is configured then it is possible to reduce the value to 2 without performance loss. Starting more than 4 threads will not improve sge_qmaster(8) scheduling and execution performance. In rare cases more than 4 threads might improve the cluster submit rate if a server JSV is configured that needs much time to verify one job.

reader_threads

The default for this parameter will be set during installation process.

The value 0 means that the reader thread pool is completely disabled in the sge_qmaster(8) process. If the reader thread pool is disabled then all read-only and read-write requests will be handled by the worker threads.

Values between 1 and 64 will enable the reader thread pool. The corresponding number of reader threads will be part of the initially created read-only thread pool when sge_qmaster(8) is started. This thread pool will then handle incoming read-only requests whereas worker threads will then handle read-write requests only.

During runtime it is not possible to enable/disable the reader thread pool. To do so it is required to adjust the bootstrap parameter and to restart sge_qmaster(8).

In systems where the reader thread pool is enabled it is possible to dynamically adjust the number of reader threads. The *qconf*(1) -at/-kt switches can be used to start/stop reader threads but any attempt to terminate the last 2 reader threads will be rejected as well as the attempt to start more that 64 threads.

scheduler_threads

The number of scheduler threads (allowed: 0-1, default set by installation: 1, off: 0). (see *qconf*(1) -kt/-at option)

jvm_threads

The number of JVM threads (allowed: 0-1, default set by installation, off: 0).

api_threads

The number of API threads (allowed: 0-1, default set by installation, off: 0).

The API threads provide a grapql API to sge_qmaster(8).

api_port

The port number the API thread is listening on for incoming http requests.

The API endpoint is `http://<qmaster_host>:<api_port>`.

otel_endpoint

The API thread can send tracing information to an opentelemetry service.

When “otel_endpoint” is configured in the format `<hostname>:<port>` then the API thread will provide tracing data to the opentelemetry service.

“otel_endpoint” is an optional parameter. If it is not configured in the bootstrap file or if its value is “none” (default) then tracing is disabled.

tls_key_pair_path

The API thread can be started in https mode instead of http if tls_key_pair_path is pointing to a directory containing valid certificates.

When "tls_key_pair_path" is configured to valid path, the API endpoint protocol will change from http to `https://<qmaster_host>:<api_port>`.

"tls_key_pair_path" is an optional parameter. If it is not configured in the bootstrap file or if its value is "none" (default) then the API endpoint remains `http://<qmaster_host>:<api_port>`

api_gentoken_auth_issuers

Specifies a comma-seperated list of accepted JWT issuers for authenticating graphql API requests. The default value for this param is set to "hpcgentoken".

api_oidc_client_id

Specify the client ID used in OpenID Connect based authentication for authenticating graphql API requests. The default value for this param is set to "none".

api_oidc_provider_url

Specify the provider URL used in OpenID Connect based authentication for authenticating graphql API requests. The default value for this param is set to "none".

streaming_threads

The number of streaming threads (allowed: 0-1, default set by installation, off: 0)

streaming_params

Defines options for the streaming thread.

They can be configured as name value pairs separated with semicolon, e.g.

```
streaming_params ENCODER=json;PRODUCER=redis
```

The following options can be configured:

Option	Description
ENCODER	Encoding used for sending events, possible values are `avro` or `json`. Default: `avro`.
PRODUCER	Streaming service used for sending events, possible values are `redis` or `kafka`. Default: `redis`.

Option	Description
PRODUCER_HOST	Host where the streaming service is running Default: `localhost`
PRODUCER_PORT	Port (TCP socket) the streaming service is listening on. Default depends on the PRODUCER setting: For redis: `6379`. For kafka: `8082`.
PREFIX	Prefix for keys, default: `sge`, will be prepended to keys, e.g. as `sge_jobs`.

COPYRIGHT

sge_calendar_conf

NAME

calendar_conf - Altair Grid Engine calendar configuration file format

DESCRIPTION

calendar_conf reflects the format of the Altair Grid Engine calendar configuration. The definition of calendars is used to specify “on duty” and “off duty” time periods for Altair Grid Engine queues on a time of day, day of week or day of year basis. Various calendars can be implemented and the appropriate calendar definition for a certain class of jobs can be attached to a queue.

calendar_conf entries can be added, modified and displayed with the -Acal, -acal, -Mcal, -mcal, -scal and -scall options to qconf(1) or with the calendar configuration dialog of the graphical user interface *qmon*(1).

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

calendar_name

The name of the calendar to be used when attaching it to queues or when administering the calendar definition. See calendar_name in sge_types(1) for a precise definition of valid calendar names.

year

The queue status definition on a day of the year basis. This field generally will specify on which days of a year (and optionally at which times on those days) a queue, to which the calendar is attached, will change to a certain state. The syntax of the **year** field is defined as follows:

```
year:=
{ NONE
| year_day_range_list=daytime_range_list[=state]
| year_day_range_list=[daytime_range_list=]state
| state}
```

Where

- NONE means, no definition is made on the year basis
- if a definition is made on the year basis, at least one of **year_day_range_list**, **daytime_range_list** and **state** always have to be present,
- all day long is assumed if **daytime_range_list** is omitted,
- switching the queue to "off" (i.e. disabling it) is assumed if **state** is omitted,
- the queue is assumed to be enabled for days neither referenced implicitly (by omitting the **year_day_range_list**) nor explicitly

and the syntactical components are defined as follows:

```
year_day_range_list := {yearday-yearday|yearday},...
daytime_range_list := hour[:minute][:second]-hour[:minute][:second],...
state := {on|off|suspended}
year_day := month_day.month.year
month_day := {1|2|...|31}
month := {jan|feb|...|dec|1|2|...|12}
year := {1970|1971|...|2037}```
```

week

The queue status definition on a day of the week basis. This field generally will specify on which days of a week (and optionally at which times on those days) a queue, to which the calendar is attached, will change to a certain state. The syntax of the **week** field is defined as follows:

```
week:=
{ NONE
| week_day_range_list[=daytime_range_list][=state]
| [week_day_range_list=]daytime_range_list[=state]
| [week_day_range_list=][daytime_range_list=]state} ...
```

Where

- NONE means, no definition is made on the week basis
- if a definition is made on the week basis, at least one of **week_day_range_list**, **daytime_range_list** and **state** always have to be present,
- every day in the week is assumed if **week_day_range_list** is omitted,
- syntax and semantics of **daytime_range_list** and **state** are identical to the definition given for the year field above,

- the queue is assumed to be enabled for days neither referenced implicitly (by omitting the **week_day_range_list**) nor explicitly

and where **week_day_range_list** is defined as

```
week_day_range_list := {weekday-weekday|weekday},...
week_day := {mon|tue|wed|thu|fri|sat|sun}
```

with week_day ranges the week_day identifiers must be different.

SEMANTICS

Successive entries to the **year** and **week** fields (separated by blanks) are combined in compliance with the following rule:

- "off"-areas are overridden by overlapping "on"- and "suspended"-areas and "suspended"-areas are overridden by "on"-areas.

Hence an entry of the form `week 12-18 tue=13-17=on` means that queues referencing the corresponding calendar are disabled the entire week from 12.00-18.00 with the exception of Tuesday between 13.00-17.00 where the queues are available.

- Area overriding occurs only within a year/week basis. If a year entry exists for a day then only the year calendar is taken into account and no area overriding is done with a possibly conflicting week area.
- the second time specification in a daytime_range_list may be before the first one and treated as expected. Thus an entry of the form `year 12.03.2004=12-11=off`causes the queue(s) be disabled 12.03.2004 from 00:00:00 - 10:59:59 and 12:00:00 - 23:59:59.

EXAMPLES

(The following examples are contained in the directory `$sge_ROOT/util/resources/calendars`).

- Night, weekend and public holiday calendar:

On public holidays "night" queues are explicitly enabled. On working days queues are disabled between 6.00 and 20.00. Saturday and Sunday are implicitly handled as enabled times:

```
calendar_name   night
year        1.1.1999,6.1.1999,28.3.1999,30.3.1999-
31.3.1999,18.5.1999-19.5.1999,3.10.1999,25.12.1999,26
.12.1999=on
week        mon-fri=6-20
```

- Day calendar:

On public holidays “day”-queues are disabled. On working days such queues are closed during the night between 20.00 and 6.00, i.e. the queues are also closed on Monday from 0.00 to 6.00 and on Friday from 20.00 to 24.00. On Saturday and Sunday the queues are disabled.

```
calendar_name   day
year        1.1.1999,6.1.1999,28.3.1999,30.3.1999-
31.3.1999,18.5.1999-19.5.1999,3.10.1999,25.12.1999,26
.12.1999
week        mon-fri=20-6 sat-sun
```

- Night, weekend and public holiday calendar with suspension:

Essentially the same scenario as the first example but queues are suspended instead of switching them “off”.

```
calendar_name   night_s
year        1.1.1999,6.1.1999,28.3.1999,30.3.1999-
31.3.1999,18.5.1999-19.5.1999,3.10.1999,25.12.1999,26
.12.1999=on
week        mon-fri=6-20=suspended
```

- Day calendar with suspension:

Essentially the same scenario as the second example but queues are suspended instead of switching them “off”.

```
calendar_name   day_s
year        1.1.1999,6.1.1999,28.3.1999,30.3.1999-
31.3.1999,18.5.1999-19.5.1999,3.10.1999,25.12.1999,26
.12.1999=suspended
week        mon-fri=20-6=suspended sat-sun=suspended
```

- Weekend calendar with suspension, ignoring public holidays:

Settings are only done on the week basis, no settings on the year basis (keyword “NONE”).

```
calendar_name   weekend_s
year        NONE
week        sat-sun=suspended
```

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), sge_queue_conf(5).

COPYRIGHT

sge_checkpoint

NAME

checkpoint - Altair Grid Engine checkpointing environment configuration file format

DESCRIPTION

Checkpointing is a facility to save the complete status of an executing program or job and to restore and restart from this so called checkpoint at a later point of time if the original program or job was halted, e.g. through a system crash.

Altair Grid Engine provides various levels of checkpointing support (see sge_ckpt(1)). The checkpointing environment described here is a means to configure the different types of checkpointing in use for your Altair Grid Engine cluster or parts thereof. For that purpose you can define the operations which have to be executed in initiating a checkpoint generation, a migration of a checkpoint to another host or a restart of a checkpointed application as well as the list of queues which are eligible for a checkpointing method.

Supporting different operating systems may easily force Altair Grid Engine to introduce operating system dependencies for the configuration of the checkpointing configuration file and updates of the supported operating system versions may lead to frequently changing implementation details. Please refer to the `<sge_root>/ckpt` directory for more information.

Please use the -ackpt, -dckpt, -mckpt or -sckpt options to the qconf(1) command to manipulate checkpointing environments from the command-line or use the corresponding qmon(1) dialogue for X-Windows based interactive configuration.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

The format of a checkpoint file is defined as follows:

ckpt_name

The name of the checkpointing environment as defined for ckpt_name in sge_types(1). qsub(1) **-ckpt** switch or for the qconf(1) options mentioned above.

interface

The type of checkpointing to be used. Currently, the following types are valid:

- *hibernator* The Hibernator kernel level checkpointing is interfaced.
- *cpr* The SGI kernel level checkpointing is used.
- *cray-ckpt* The Cray kernel level checkpointing is assumed.
- *transparent* Altair Grid Engine assumes that the jobs submitted with reference to this checkpointing interface use a checkpointing library such as provided by the public domain package Condor.
- *userdefined* Altair Grid Engine assumes that the jobs submitted with reference to this checkpointing interface perform their private checkpointing method.
- *application-level* Uses all of the interface commands configured in the checkpointing object like in the case of one of the kernel level checkpointing interfaces (cpr, cray-ckpt, etc.) except for the **restart_command** (see below), which is not used (even if it is configured) but the job script is invoked in case of a restart instead.

ckpt_command

A command-line type command string to be executed by Altair Grid Engine in order to initiate a checkpoint.

migr_command

A command-line type command string to be executed by Altair Grid Engine during a migration of a checkpointing job from one host to another.

restart_command

A command-line type command string to be executed by Altair Grid Engine when restarting a previously checkpointed application.

clean_command

A command-line type command string to be executed by Altair Grid Engine in order to cleanup after a checkpointed application has finished.

ckpt_dir

A file system location to which checkpoints of potentially considerable size should be stored.

ckpt_signal

A Unix signal to be sent to a job by Altair Grid Engine to initiate a checkpoint generation. The value for this field can either be a symbolic name from the list produced by the -l option of the kill(1) command or an integer number which must be a valid signal on the systems used for checkpointing.

when

The points of time when checkpoints are expected to be generated. Valid values for this parameter are composed by the letters s, m, x and r and any combinations thereof without any separating character in between. The same letters are allowed for the -c option of the qsub(1) command which will overwrite the definitions in the used checkpointing environment. The meaning of the letters is defined as follows:

Value	Description
s	A job is checkpointed, aborted and if possible migrated if the corresponding sge_execd(8) is shut down on the job's machine.
m	Checkpoints are generated periodically at the min_cpu_interval interval defined by the queue (see sge_queue_conf(5)) in which a job executes.
x	A job is checkpointed, aborted and if possible migrated as soon as the job gets suspended (manually as well as automatically).
r	A job will be rescheduled (not checkpointed) when the host on which the job currently runs went into unknown state and the time interval reschedule_unknown (see sge_conf(5)) defined in the global/local cluster configuration will be exceeded.

RESTRICTIONS

Note, that the functionality of any checkpointing, migration or restart procedures provided by default with the Altair Grid Engine distribution as well as the way how they are invoked in the ckpt_command, migr_command or restart_command parameters of any default checkpointing environments should not be changed or otherwise the functionality remains the full responsibility of the administrator configuring the checkpointing environment. Altair Grid Engine will just invoke these procedures and evaluate their exit status. If the procedures do not perform their tasks properly or are not invoked in a proper fashion, the checkpointing mechanism may behave unexpectedly, Altair Grid Engine has no means to detect

this.

SEE ALSO

sge_intro(1), sge_ckpt(1), sge_types(1), qconf(1), qmod(1), qsub(1), sge_execd(8).

COPYRIGHT

sge_complex

NAME

complex - Altair Grid Engine complexes configuration file format

DESCRIPTION

Complex reflects the format of the Altair Grid Engine complex configuration. The definition of complex attributes provides all pertinent information concerning the resource attributes a user may request for a Altair Grid Engine job via the qsub(1) **-l** option and for the interpretation of these parameters within the Altair Grid Engine system.

The Altair Grid Engine complex object defines all entries which are used for configuring the global, the host, and queue object. The system has a set of pre defined entries, which are assigned to a host or queue per default. In a addition can the user define new entries and assign them to one or multiple objects. Each load value has to have its corresponding complex entry object, which defines the type and the relational operator for it.

Defining resource attributes

The complex configuration should not be accessed directly. In order to add or modify complex entries, the qconf(1) options -Mc and -mc should be used instead. While the -Mc option takes a complex configuration file as an argument and overrides the current configuration, the -mc option bring up an editor filled in with the current complex configuration.

The provided list contains all definitions of resource attributes in the system. Adding a new entry means to provide: name, shortcut, type, relop, requestable, consumable, default, and urgency. The fields are described below. Changing one is easily done by updating the field to change and removing an entry by deleting its definition. An attribute can only be removed, when it is not referenced in a host or queue object anymore. Also does the system have a set of default resource attributes which are always attached to a host or queue. They cannot be deleted nor can the type of such an attribute be changed.

Working with resource attributes

Before a user can request a resource attribute it has to be attached to the global, host, or cqueue object. The resource attribute exists only for the objects, it got attached to (if it is attached to the global object(qconf -me global), it exits system wide, host object: only on that host (qconf -me NAME): cqueue object: only on that cqueue (qconf -mq NAME)).

When the user attached a resource attribute to an object, one also has to assign a value to it; the resource limit. Another way to get a resource attribute value is done by configuring a load sensor for that attribute.

Default queue resource attributes

In its default form it contains a selection of parameters in the queue configuration as defined in sge_queue_conf(5). The queue configuration parameters being requestable for a job by the user in principal are:

- qname The qname where the job shoul run in
- hostname The hostname of the requested host
- notify The request to notify a job about a soft/hard resource limit
- calendar requesting a queue, providing a specific calendar
- min_cpu_interval requesting a queue, providing a specific cpu_interval
- tmpdir requesting a queue, providing a specific tmpdir setup
- seq_no requesting a queue, providing a specific seq_no
- s_rt requesting a queue, providing a specific soft limit
- h_rt requesting a queue, providing a specific hard limit
- s_cpu "
- h_cpu "
- s_data "
- h_data "
- s_stack "
- h_stack "
- s_core "

- h_core "
- s_rss "
- h_rss "

Default host resource attributes

The standard set of host related attributes consists of two categories. The first category is built by several queue configuration attributes which are particularly suitable to be managed on a host basis. These attributes are:

- slots requesting a queue or a host providing specific amount of free slots (parallel jobs -pe)

- s_vmem requesting a queue, providing a specific soft vmem limit
- h_vmem requesting a queue, providing a specific hard vmem limit
- s_fsize "
- h_fsize "

(please refer to sge_queue_conf(5) for details).

Additional builtin requestable resources

The standard set of resources, set by default as requestable and usually used to request a for the job required resource. Usually requested qith qsub -l requests These resources

are initialized via the execution nodes architecture, cpu types, mounted memory, other installed hard- and software (eg. GPU cards, installed docker daemon) The values of the resources are initialized with detected values during installation and/or startup and via overriding them via the complex_values settings in the execution host configuration. (please refer to sge_host_conf(5) for details)

- arch
- breeze
- cpu
- display_win_gui
- docker
- docker_api_version
- docker_images
- docker_version
- m_cache_l1
- m_cache_l2
- m_cache_l3
- m_core
- m_gpu
- m_mem_free
- m_mem_free_n0
- m_mem_free_n1
- m_mem_free_n2
- m_mem_free_n3
- m_mem_total
- m_mem_total_n0
- m_mem_total_n1
- m_mem_total_n2
- m_mem_total_n3
- m_mem_used
- m_mem_used_n0
- m_mem_used_n1
- m_mem_used_n2
- m_mem_used_n3
- m_numa_nodes
- m_socket
- m_thread
- m_topology
- m_topology_inuse
- m_topology_numa
- mem_free
- mem_total
- mem_used
- mistral
- num_proc
- swap_free
- swap_rate
- swap_rsvd
- swap_total

- swap_used
- virtual_free
- virtual_total
- virtual_used

Additional builtin non-requestable resources

The standard set of resources, set by default as not requestable and usually used to report load or requested by an explicit qsub switch, eg. qsub -r yes for rerunable jobs

- load_avg
- load_long
- load_medium
- load_short
- np_load_avg

- np_load_long

- np_load_medium
- np_load_short
- rerun

Note: Defining these attributes in the host complex is no contradiction to having them also in the queue configuration. It allows maintaining the corresponding resources on a host level and at the same time on a queue level. Total virtual free memory (h_vmem) can be managed for a host, for example, and a subset of the total amount can be associated with a queue on that host.

The second attribute category in the standard host complex are the default load values Every sge_execd(8) periodically reports load to sge_qmaster(8). The reported load values are either the standard Altair Grid Engine load values such as the CPU load average (see uptime(1)) or load values defined by the Altair Grid Engine administration (see the **load_sensor** parameter in the cluster configuration sge_conf(5) and the Altair Grid Engine Installation and Administration Guide for details). The characteristics definition for the standard load values is part of the default host complex, while administrator defined load values require extension of the host complex. Please refer to the file `<sge_root>/doc/load_parameters.asc` for detailed information on the standard set of load values.

Overriding attributes

One attribute can be assigned to the global object, host object, and queue object at the same time. On the host level it might get its value from the user defined resource limit and a load sensor. In case that the attribute is a consumable, we have in addition to the resource limit and its load report on host level also the internal usage, which the system keeps track of. The merge is done as follows:

In general an attribute can be overridden on a lower level

- global by hosts and queues
- hosts by queues and load values or resource limits on the same level.

We have one limitation for overriding attributes based on its relational operator:

!=, == operators can only be overridden on the same level, but not on a lower level. The user defined value always overrides the load value.

>=, >, <=, < operators can only be overridden, when the new value is more restrictive than the old one.

In the case of a consumable on host level, which has also a load sensor, the system checks for the current usage, and if the internal accounting is more restrictive than the load sensor report, the internal value is kept; if the load sensor report is more restrictive, that one is kept.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

The principal format of a *complex* configuration is that of a tabulated list. Each line starting with a '#' character is a comment line. Each line despite comment lines define one element of the complex. A element definition line consists of the following 8 column entries per line (in the order of appearance):

name

The name of the complex element to be used to request this attribute for a job in the qsub(1) **-l** option. A complex attribute name (see complex_name in sge_types(1)) may appear only once across all complexes, i.e. the complex attribute definition is unique.

shortcut

A shortcut for **name** which may also be used to request this attribute for a job in the qsub(1) **-l** option. An attribute **shortcut** may appear only once across all complexes, so as to avoid the possibility of ambiguous complex attribute references.

type

This setting determines how the corresponding values are to be treated Altair Grid Engine internally in case of comparisons or in case of load scaling for the load complex entries:

- With **INT** only raw integers are allowed.

- With **DOUBLE** floating point numbers in double precision (decimal and scientific notation) can be specified.
- With **TIME** time specifiers are allowed. Refer to sge_queue_conf(5) for a format description.
- With **MEMORY** memory size specifiers are allowed. Refer to sge_queue_conf(5) for a format description.
- With **BOOL** the strings TRUE and FALSE are allowed. When used in a load formula (refer to *sge_sched_conf*(5)) TRUE and FALSE get mapped into '1' and '0'.
- With **STRING** all strings are allowed and is used for wildcard regular boolean expression matching. Please see *sge_types*(1) manpage for **expression** definition.
- With **RSMAP** integers and strings are allowed. See sge_resource_map(5) for a format description.

Examples:

-l arch="*x*|sol*" :
results in "arch=lx-x86" OR "arch=lx-amd64"
OR "arch=sol-amd64" OR ...
-l arch="sol-x??" :
results in "arch=sol-x86" OR "arch=sol-x64" OR ...
-l arch="lx2[246]-x86" :
results in "arch=lx22-x86" OR "arch=lx24-x86"
OR "arch=lx26-x86"
-l arch="lx2[4-6]-x86" :
results in "arch=lx24-x86" OR "arch=lx25-x86"
OR "arch=lx26-x86"
-l arch="lx2[24-6]-x86" :
results in "arch=lx22-x86" OR "arch=lx24-x86"
OR "arch=lx25-x86" OR "arch=lx26-x86"
-l arch="!lx-x86&!sol-amd64" :
results in NEITHER "arch=lx-x86" NOR "arch=sol-amd64"
-l arch="lx2[4|6]-amd64" :
results in "arch=lx24-amd64" OR "arch=lx26-amd64"

- **CSTRING** is like **STRING** except comparisons are case insensitive.
- **RESTRING** is like **STRING** and it will be deprecated in the future.
- **HOST** is like **CSTRING** but the expression must match a valid hostname.
- For **INT**, **DOUBLE**, **MEMORY** and **RSMAP** consumable requests it is possible to specify a range (soft_range) if the consumable is requested by using the **-soft** option. Please see sge_types(1) manpage for **soft_range** definition. If a soft request is not grantable the job will be dispatched anyway instead of staying in the pending job list - the request is simply ignored. Jobs that are requesting resources within the **-hard** section of the submit command have to wait until there are enough resources free and will stay in the pending job list.

Examples:

1) qsub … -soft -l memory=4K-4G:4K …

Soft request for consumable “memory”. Any value between 4096 Byte and 4294967296 Byte that matches the specified step of 4096 Byte will be granted if it is available on the resulting hosts which were selected by the scheduler run. The scheduler will select hosts based on the specified **-hard** requested resources and on the configured scheduler settings.

2) qsub … -soft -l “memory=4G-8G:1G|1G-4G:500M” …

Soft request with 2 ranges for consumable “memory”. If there is no resource available that provides the big memory range (4G-8G) the algorithm will try to find a matching resource for the 2nd specified range.

It is also possible to request some minimum value for the consumable in the **-hard** section and define additionally a higher value or range in the **-soft** part of the request.

Examples:

1) qsub … -soft -l my_double=2.5-8.5:0.1 -hard -l my_double=2.5 …

Soft request with ranges for consumable “my_double” in combination with hard request. The job will stay pending until there is a minimum of my_double=2.5 available on a host or queue. Once a resource was found the consumable might also consume up to 8.5 - using a range step size of 0.1.

1) qsub … -soft -l my_int=“20-100:2|10-20:1|1-10:1”,arch=lx-amd64 -hard -l my_int=1 …

Soft request with 3 ranges for consumable “my_int” and preferred architecture “lx-amd64” in combination with hard request. The job will stay pending until there is a minimum of my_int=1 available on a host or queue. In addition to the “my_int” consumable also the architecture “lx_amd64” is preferred.

relop

The **relation operator.** The relation operator is used when the value requested by the user for this parameter is compared against the corresponding value configured for the considered queues. If the result of the comparison is false, the job cannot run in this queue. Possible relation operators are “==”, “<”, “>”, “<=”, “>=” and “EXCL”. The only valid operator for string type attributes is “==”.

The “EXCL” relation operator implements exclusive scheduling and is only valid for consumable boolean type attributes. Exclusive means the result of the comparison is only true if a job requests to be exclusive and no other exclusive or non-exclusive jobs uses the complex. If the job does not request to be exclusive and no other exclusive job uses the complex the comparison is also true.

requestable

The entry can be used in a *qsub*(1) resource request if this field is set to 'y' or 'yes'. If set to 'n' or 'no' this entry cannot be used by a user in order to request a queue or a class of queues. If the entry is set to 'forced' or 'f' the attribute has to be requested by a job or it is rejected.

To enable resource request enforcement the existence of the resource has to be defined. This can be done on a cluster global, per host and per queue basis. The definition of resource availability is performed with the complex_values entry in sge_host_conf(5) and sge_queue_conf(5).

consumable

The **consumable** parameter can be set to either 'yes' ('y' abbreviated), 'no' ('n'), 'JOB' ('j'), or 'HOST' ('h'). It can be set to 'yes' and 'JOB' only for numeric attributes (INT, DOUBLE, MEMORY, TIME, RSMAP - see **type** above). It can be set to 'HOST' only for a RSMAP attribute (which must be initialized on host layer). If set to 'yes', 'JOB' or 'HOST' the consumption of the corresponding resource can be managed by Altair Grid Engine internal bookkeeping. In this case Altair Grid Engine accounts for the consumption of this resource for all running jobs and ensures that jobs are only dispatched if the Altair Grid Engine internal bookkeeping indicates enough available consumable resources. Consumables are an efficient means to manage limited resources such a available memory, free space on a file system, network bandwidth or floating software licenses.

A consumable defined by 'y' is a per slot consumable which means the limit is multiplied by the number of slots being used by the job before being applied. In case of 'j' the consumable is a per job consumable. This resource is debited as requested (without multiplication) from the allocated master queue. The resource needs not be available for the slave task queues. If slave tasks explicitely request that resource, the request is ignored.

A consumable defined as 'HOST' is a per host consumable which means that the request is not multiplied by the number of tasks on that host. In case a parallel job spans multiple hosts, the resource is requested just one time on each host. If multiple parallel tasks on the same host explicitely request that resource, it is granted to each task, but only the maximum request on that host is debited from the cluster. Use cases are parallel jobs which are requesting co-processor cards, like GPUs or other hardware. Regardless of how many parallel tasks finally run on a host, just the exact amount of the (not multiplied by slots) is consumed.

Consumables can be combined with default or user defined load parameters (see sge_conf(5) and sge_host_conf(5)), i.e. load values can be reported for consumable attributes or the consumable flag can be set for load attributes. The Altair Grid Engine consumable resource management takes both the load (measuring availability of the resource) and the internal bookkeeping into account in this case, and makes sure that neither of both exceeds a given limit.

To enable consumable resource management the basic availability of a resource has to be defined. This can be done on a cluster global, per host and per queue instance basis while these categories may supersede each other in the given order (i.e. a host can restrict availability of a cluster resource and a queue can restrict host and cluster resources). The definition of resource availability is performed with the **complex_values** entry in sge_host_conf(5)

and sge_queue_conf(5). The **complex_values** definition of the "global" host specifies cluster global consumable settings. To each consumable complex attribute in a **complex_values** list a value is assigned which denotes the maximum available amount for that resource. The internal bookkeeping will subtract from this total the assumed resource consumption by all running jobs as expressed through the jobs' resource requests.

Note: Jobs can be forced to request a resource and thus to specify their assumed consumption via the 'force' value of the **requestable** parameter (see above).

Note also: A default resource consumption value can be pre-defined by the administrator for consumable attributes not explicitly requested by the job (see the **default** parameter below). This is meaningful only if requesting the attribute is not enforced as explained above.

See the Altair Grid Engine Installation and Administration Guide for examples on the usage of the consumable resources facility.

default

Meaningful only for consumable complex attributes (see **consumable** parameter above). Altair Grid Engine assumes the resource amount denoted in the **default** parameter implicitly to be consumed by jobs being dispatched to a host or queue managing the consumable attribute. Jobs explicitly requesting the attribute via the -l option to qsub(1) override this default value.

Note: Setting a default value is not supported for complexes with type RSMAP (see sge_resource_map(5)).

urgency

The urgency value allows influencing job priorities on a per resource base. The urgency value effects the addend for each resource when determining the resource request related urgency contribution. For numeric type resource requests the addend is the product of the urgency value, the jobs assumed slot allocation and the per slot request as specified via -l option to qsub(1). For string type requests the resources urgency value is directly used as addend. Urgency values are of type real. See under sge_priority(5) for an overview on job priorities.

aapre

The aapree-attribute (available after preemption) of a complex defines if a resource will be reported as available within Altair Grid Engine when a job that consumes such a resource is preempted. For all non-consumable resources it can only be set to NO. For consumables it can be set to YES or NO. The aapre-attribute of the slots complex can only be set to YES. After the installation of Altair Grid Engine all memory based complexes are defined as consumable and aapre is also set to YES. As result the system will report memory (and slots) as available that are in the preempted state (P-state).

Note: Please note that there are different preemptive actions available that can 'preempt' a job. Depending on the preemptive action that is applied to a job, this job might switch into

P-state, N-state or S-state. For jobs in P-state all used resources where the aapre-attribute is set to true will be reported as available. For the N-state only all non-memory based consumables where aapre-attribute is true. For the S-state only the slots consumable will be available.

Find more information concerning preemptive actions and resulting job states in sge_preemption(5).

affinity

Affinity is a double value that influences the attraction/rejection of corresponding jobs requesting that resource and therefore allows to implement affinity and anti-affinity as job placement policy beginning with Altair Grid Engine 8.6.0. Default value for all default complexes as well as for new complexes that are created is 0.0. Jobs requesting resources with a positive affinity value attract each other whereas negative values cause rejection of corresponding jobs requesting that resource.

For all resource types (also non-number based complexes like restring) the absolute number of the affinity will be used as affinity for the corresponding resource request of a job. For consumable resources the affinity value of a resource will act as multiplier for the underlying resource requests of a job that are granted. This means that one big running job attracts/rejects to the same extent as 'multiple small' running jobs within the same host or queue as long as the 'big' and 'multiple small' jobs consume the same amount of resources.

In case of multiple resource requests of complex attributes with non-zero affinity setting the job's affinity value is the sum of affinity values of corresponding resources. The sum of affinity values of all jobs already running on a host/queue cause attraction/rejection of corresponding jobs in the pending job list. Depending on the active scheduling policies and on weighting between them this will cause affinity (so that jobs build groups on nodes), anti-affinity (so that jobs are distributed on host in the cluster or queue residing on hosts) or best fit.

do_report

The do_report attribute defines if a resource may be reported as a load value by sge_execd(8).

When it is set to YES and sge_execd(8) or a load sensor running on an execution host gathers information about the resource, load values will be sent by sge_execd(8) to sge_qmaster(8) and will be reported by 'qstat(1) -F', 'qhost(1) -F' and 'qconf(1) -se <hostname>'.

When do_report is set to NO then no load values will be reported, even if a load sensor would gather information about a resource. Exception are all "m_mem_*" and all "cuda.*" complex variables which will always be reported.

Especially in big clusters it can make sense to disable load reporting of individual resources by setting do_report to NO. Please note that for certain builtin resources like arch, num_proc, m_socket, m_core etc. load reporting cannot be disabled.

is_static

The is_static attribute allows to define if the load value for a resource is considered being static (does not or seldom change) or dynamic (constantly changes).

Examples for static load values are arch, m_socket, m_core. These values are available if they were reported once by sge_execd(8), even if sge_execd(8) is currently not running or the host is down.

Non-static load values are dynamically changing resources like load_avg, np_load_avg, mem_used, mem_free. When a host and/or the sge_execd(8) on a host is down no values for such resources are available.

SEE ALSO

sge_intro(1), sge_types(1), sge_preemption(5), sge_resource_map(5), qconf(1), qsub(1), uptime(1), sge_host_conf(5), sge_queue_conf(5), sge_execd(8), sge_qmaster(8), Altair Grid Engine Installation and Administration Guide.

COPYRIGHT

sge_conf

NAME

sge_conf - Altair Grid Engine configuration files

DESCRIPTION

sge_conf defines the global and local Altair Grid Engine configurations and can be shown/modified using qconf(1) with the -sconf/-mconf options. Only root or the cluster administrator may modify sge_conf.

At its initial start-up, sge_qmaster(8) checks to see whether a valid Altair Grid Engine configuration is available at a well-known location in the Altair Grid Engine internal directory hierarchy. If so, it loads that configuration information and proceeds. If not, sge_qmaster(8) writes a generic configuration containing default values to that same location. The Altair Grid Engine execution daemons sge_execd(8) upon start-up retrieve their configuration from sge_qmaster(8).

The actual configuration for both sge_qmaster(8) and sge_execd(8) is a superposition of a global configuration and a local configuration for the host on which a master or execution daemon resides. If a local configuration is available, its entries overwrite the corresponding entries of the global configuration. **Note:** The local configuration does not have to contain all possible configuration entries, but only those which need to be modified against the global entries.

Note: Altair Grid Engine allows backslashes (\) to be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

The paragraphs that follow provide brief descriptions of the individual parameters in the global and local configurations for a Altair Grid Engine cluster:

execd_spool_dir

The execution daemon spool directory path. This spool directory requires read/write access permission for root. The entry in the global configuration for this parameter can be overwritten by execution host local configurations, i.e. each sge_execd(8) may have a private spool directory with its own path, in which case it needs to provide read/write permission for the root account of the corresponding execution host only.

Under **execd_spool_dir** a directory whose name corresponds to the unqualified hostname of the execution host is opened and contains all information spooled to disk. Thus, it is possible for the **execd_spool_dir**s of all execution hosts to physically reference the same directory path (the root access restrictions mentioned above need to be met, however).

Changing the global **execd_spool_dir** parameter set at installation time is not supported in a running system. If the change should still be done, it is required to restart all affected execution daemons. Please make sure running jobs have finished before doing so, otherwise running jobs will be lost.

The default location for the execution daemon spool directory is '$sge_ROOT/$sge_CELL/spool'.

The global configuration entry for this value may be overwritten by the execution host local configuration.

mailer

mailer is the absolute pathname to the electronic mail delivery agent on your system. It must accept the following syntax:

mailer -s

Each sge_execd(8) may use a private mail agent. Changing **mailer** will take immediate effect.

The default for **mailer** depends on the operating system of the host on which the Altair Grid Engine master installation was run. Common values are /bin/mail or /usr/bin/Mail.

The global configuration entry for this value may be overwritten by the execution host local configuration.

xterm

xterm is the absolute pathname to the X Window System terminal emulator, xterm(1).

Changing **xterm** will take immediate effect.

The default for **xterm** is /usr/bin/X11/xterm.

The global configuration entry for this value may be overwritten by the execution host local configuration.

load_sensor

A comma-separated list of executable shell script paths or programs to be started by sge_execd(8) and to be used in order to retrieve site configurable load information (e.g. free space on a certain disk partition).

Each sge_execd(8) may use a set of private **load_sensor** programs or scripts. Changing **load_sensor** will take effect after two load report intervals (see **load_report_time**). A load sensor will be restarted automatically if the file modification time of the load sensor executable changes.**

The global configuration entry for this value may be overwritten by the execution host local configuration.

In addition to the load sensors configured via **load_sensor**, sge_execd(8) searches for an executable file named qloadsensor in the execution host's Altair Grid Engine binary directory path. If such a file is found, it is treated like the configurable load sensors defined in **load_sensor**. This facility is intended for pre-installing a default load sensor.

prolog

The executable path of a shell script that is started before execution of Altair Grid Engine jobs with the same environment setting as that for the Altair Grid Engine jobs to be started afterwards. An optional prefix "user@" specifies the user under which this procedure is to be started. The procedure's standard output and the error output stream are written to the same file used for the standard output and error output of each job. This procedure is intended as a means for the Altair Grid Engine administrator to automate the execution of general site-specific tasks, such as the preparation of temporary file systems, that need the same context information as the job. Each sge_execd(8) may use a private prolog script. Correspondingly, the execution host local configuration can be overwritten by the queue configuration (see sge_queue_conf(5)). Changing **prolog** will take immediate effect.

The default for **prolog** is the special value NONE, which prevents the execution of a prolog script.

Scripts whose execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_params**.

The following special variables expanded at runtime can be used (in addition to any other strings which have to be interpreted by the procedure) to constitute a command line:

Value	Description
`$host`	The name of the host on which the prolog or epilog procedures are started.
`$job_owner`	The user name of the job owner.
`$job_id`	Altair Grid Engine's unique job identification number.
`$job_name`	The name of the job.
`$processors`	The "processors" string as contained in the queue configuration (see *sge_queue_conf*(5)) of the master queue (the queue in which the prolog and epilog procedures are started).
`$queue`	The cluster queue name of the master queue instance, i.e. the cluster queue in which the prolog and epilog procedures are started.
`$stdin_path`	The pathname of the stdin file. This is always /dev/null for prolog, pe_start, pe_stop and epilog. It is the pathname of the stdin file for the job in the job script. When delegated file staging is enabled, this path is set to `$fs_stdin_tmp_path`. When delegated file staging is not enabled, it is the stdin pathname given via DRMAA or qsub.

Value	Description
`$stdout_path`	The pathname of the stdout/stderr file. This always points to the output/error file. When delegated file staging is enabled, this path is set to `$fs_stdout_tmp_path/$fs_stderr_tmp_path`. When delegated file staging is not enabled, it is the stdout/stderr pathname given via DRMAA or qsub.
`$stderr_path`	
`$merge_stderr`	If merging of stderr and stdout is requested, this flag is "1", otherwise it is "0". If this flag is 1, stdout and stderr are merged in one file, the stdout file. Merging of stderr and stdout can be requested via the DRMAA job template attribute *drmaa_join_files* (see *drmaa_attributes*(3)) or the qsub parameter *-j y* (see *qsub*(1)).
`$fs_stdin_host`	When delegated file staging is requested for the stdin file, this is the name of the host where the stdin file has to be copied from before the job is started.
`$fs_stdout_host`	When delegated file staging is requested for the stdout/stderr file, this is the name of the host where the stdout/stderr file has to be copied to after the job has run.
`$fs_stderr_host`	
`$fs_stdin_path`	When delegated file staging is requested for the stdin file, this is the pathname of the stdin file on the host `$fs_stdin_host`.
`$fs_stdout_path`	When delegated file staging is requested for the stdout/stderr file, this is the pathname of the stdout/stderr file on the host `$fs_stdout_host/$fs_stderr_host`.
`$fs_stderr_path`	
`$fs_stdin_tmp_path`	When delegated file staging is requested for the stdin file, this is the destination pathname of the stdin file on the execution host. The prolog script must copy the stdin file from `$fs_stdin_host:$fs_stdin_path` to `localhost:$fs_stdin_tmp_path` to establish delegated file staging of the stdin file.
`$fs_stdout_tmp_path`	When delegated file staging is requested for the stdout/stderr file, this is the source pathname of the stdout/stderr file on the execution host. The epilog script must copy the stdout file from `localhost:$fs_stdout_tmp_path` to `$fs_stdout_host:$fs_stdout_path` (the stderr file from `localhost:$fs_stderr_tmp_path` to `$fs_stderr_host:$fs_stderr_path`) to establish delegated file staging of the stdout/stderr file.
`$fs_stderr_tmp_path`	
`$fs_stdin_file_staging`	When delegated file staging is requested for the stdin/stdout/stderr file, the flag is set to "1", otherwise it is set to "0" (see in **delegated_file_staging** how to enable delegated file staging).
`$fs_stdout_file_staging`	
`$fs_stderr_file_staging`	

These three flags correspond to the DRMAA job template attribute 'drmaa_transfer_files' (see drmaa_attributes(3)).

The global configuration entry for this value may be overwritten by the execution host local configuration.

Exit codes for the prolog attribute can be interpreted based on the following exit values:

exit code	description
0	Success
99	Reschedule job
100	Put job in error state
Anything else	Put queue in error state

epilog

The executable path of a shell script that is started after execution of Altair Grid Engine jobs with the same environment setting as that for the Altair Grid Engine jobs that has just completed. An optional prefix "user@" specifies the user under which this procedure is to be started. The procedures standard output and the error output stream are written to the same file used also for the standard output and error output of each job. This procedure is intended as a means for the Altair Grid Engine administrator to automate the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job. Each sge_execd(8) may use a private epilog script. Correspondingly, the execution host local configurations is can be overwritten by the queue configuration (see sge_queue_conf(5)). Changing **epilog** will take immediate effect.

The default for **epilog** is the special value NONE, which prevents from execution of a epilog script. The same special variables as for **prolog** can be used to constitute a command line.

Scripts where the execution duration would exceed 2 minutes will be terminated. This time-out can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_params**.

The global configuration entry for this value may be overwritten by the execution host local configuration.

Exit codes for the epilog attribute can be interpreted based on the following exit values:

exit code	description
0	Success
99	Reschedule job
100	Put job in error state
Any other value <= 127	Put queue in error state, re-queue the job
Any value > 127	If **RESCHEDULE_ON_KILLED_EPILOG** is set to "true" or "1", the queue is put in an error state, and the job is re-queued. If this parameter is set to "false" or "0", the job simply finishes.

shell_start_mode

Note: Deprecated, may be removed in future release.
This parameter defines the mechanisms which are used to actually invoke the job scripts on the execution hosts. The following values are recognized:

unix_behavior If a user starts a job shell script under UNIX interactively by invoking it with just the script name, the operating system's executable loader uses the information provided in a comment such as '#!/bin/csh' in the first line of the script to detect which command interpreter to start to interpret the script. This mechanism is used by Altair Grid Engine when starting jobs if unix_behavior is specified as **shell_start_mode**.

posix_compliant POSIX does not consider first script line comments such as '#!/bin/csh' to be significant. The POSIX standard for batch queuing systems (P1003.2d) therefore requires a compliant queuing system to ignore such lines but to use user-specified or configured default command interpreters instead. Thus, if **shell_start_mode** is set to posix_compliant Altair Grid Engine will either use the command interpreter indicated by the **-S** option of the qsub(1) command or the **shell** parameter of the queue to be used (see sge_queue_conf(5) for details).

script_from_stdin Setting the **shell_start_mode** parameter to either posix_compliant or unix_behavior requires you to set the umask in use for sge_execd(8) such that every user has read access to the active_jobs directory in the spool directory of the corresponding execution daemon. If you have **prolog** and **epilog** scripts configured, they also need to be readable by any user who may execute jobs.
If this violates your site's security policies you may want to set **shell_start_mode** to script_from_stdin. This will force Altair Grid Engine to open the job script as well as the epilog and prolog scripts for reading into STDIN as root (if sge_execd(8) was started as root) before changing to the job owner's user account. The script is then fed into the STDIN stream of the command interpreter indicated by the **-S** option of the qsub(1) command or the **shell** parameter of the queue to be used (see sge_queue_conf(5) for details).
Therefore setting **shell_start_mode** to *script_from_stdin* also implies posix_compliant behavior. **Note**, however, that feeding scripts into the STDIN stream of a command interpreter may cause trouble if commands like rsh(1) are invoked inside a job script as they also process the STDIN stream of the command interpreter. These problems can usually be resolved by redirecting the STDIN channel of those commands to come from /dev/null (e.g. rsh host date < /dev/null). **Note also**, that any command-line options associated with the job are passed to the executing shell. The shell will only forward them to the job if they are not recognized as valid shell options.

Changes to **shell_start_mode** will take immediate effect. The default for **shell_start_mode** is unix_behavior.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

login_shells

UNIX command interpreters like the Bourne-Shell (see sh(1)) or the C-Shell (see csh(1)) can be used by Altair Grid Engine to start job scripts. The command interpreters can either be started as login-shells (i.e. all system and user default resource files like .login or .profile

will be executed when the command interpreter is started and the environment for the job will be set up as if the user has just logged in) or just for command execution (i.e. only shell specific resource files like .cshrc will be executed and a minimal default environment is set up by Altair Grid Engine - see qsub(1)). The parameter **login_shells** contains a comma-separated list of the executable names of the command interpreters to be started as login-shells. Shells in this list are only started as login shells if the parameter **shell_start_mode** (see above) is set to posix_compliant.

Changes to **login_shells** will take immediate effect. The default for **login_shells** is "sh,bash,csh,tcsh,ksh".

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

min_uid

min_uid** places a lower bound on user IDs that may use the cluster. Users whose user ID (as returned by getpwnam(3)) is less than **min_uid** will not be allowed to run jobs on the cluster.

Changes to **min_uid** will take immediate effect. The default for **min_uid** is 0.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

min_gid

This parameter sets the lower bound on group IDs that may use the cluster. Users whose default group ID (as returned by getpwnam(3)) is less than **min_gid** will not be allowed to run jobs on the cluster.

Changes to **min_gid** will take immediate effect. The default for **min_gid** is 0.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

user_lists

The **user_lists** parameter contains a comma-separated list of user access lists as described in sge_access_list(5). Any user listed in at least one of these access lists has access to the cluster. If the **user_lists** parameter is set to NONE (the default), any user has access who is not explicitly excluded via the **xuser_lists** parameter described below. If a user is listed in both an access list in **xuser_lists** and an access list in **user_lists**, the user is denied access to the cluster.

Changes to **user_lists** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

xuser_lists

The **xuser_lists** parameter contains a comma-separated list of user access lists as described in *sge_access_list*(5). Each user listed in at least one of these access lists is denied access to the cluster. If the **xuser_lists** parameter is set to NONE (the default), any user has access. If a user is listed in both an access list in **xuser_lists** and in an access list in **user_lists** (see above), the user is denied access to the cluster.

Changes to **xuser_lists** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

administrator_mail

administrator_mail specifies a comma-separated list of the electronic mail address(es) of the cluster administrator(s) to whom internally-generated problem reports are sent. The mail address format depends on your electronic mail system and how it is configured; consult your system's configuration guide for more information.

Changing **administrator_mail** takes immediate effect. The default for **administrator_mail** is an empty mail list.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

projects

The **projects** list contains all projects to which a job can be submitted or to which a pending job can be altered. If the **projects** list is defined, only jobs which are submitted or altered to one of these projects are accepted by Altair Grid Engine; all other jobs are rejected. If the **projects** list is not defined (i.e. it is "none"), jobs are not rejected because of their project membership.

Changing **projects** takes immediate effect. Changing **projects** doesn't affect pending or running jobs, unless the user alters a job's project, in which case the job must conform to the project requirement. The default for **projects** is none.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

xprojects

The **xprojects** list contains all projects to which a job cannot be submitted and to which a pending job cannot be altered. If the **xprojects** list is defined, all jobs that are submitted or altered to one of these projects are rejected by Altair Grid Engine; all other jobs are not rejected. If the **xprojects** list is not defined (i.e. it is "none"), jobs are not rejected because of the project membership.

Changing **xprojects** takes immediate effect. Changing **xprojects** doesn't affect pending or running jobs, except for altering them. The default for **xprojects** is none.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

load_report_time

The parameter **load_report_time** defines the time interval between load reports. Load reports are periodically reported by the execution daemons to sge_qmaster(8). The time format specification is “hh:mm:ss” (Hours:Minutes:Seconds) or just a positive integer which is interpreted as a value in seconds. The default for **load_report_time** is 40 seconds (“00:00:40”).

Each sge_execd(8) may use a different load report time. Changing **load_report_time** will take immediate effect.

Note: Be careful when modifying **load_report_time**. Reporting load too frequently might block sge_qmaster(8) especially if the number of execution hosts is large. Moreover, since the system load typically increases and decreases smoothly, frequent load reports hardly offer any benefit.

The global configuration entry for this value may be overwritten by the execution host local configuration.

gdi_request_limits

This parameter can be used to define a maximum number of requests per second that sge_qmaster(8) will accept before it starts rejecting incoming requests. The value NONE, which is the default for this parameter, means that all valid requests that will be received by the sge_qmaster(8) process will also be accepted, processed, and answered. Incoming requests that are accepted and that cannot immediately be answered will be stored in request queues until a thread is available to handle the request and send a response to the client.

Instead of NONE, a comma-separated list of limit rules can be specified. A limit rule consist of a set of filters and a number that expresses how many requests per second are allowed for those requests that match the corresponding filters. There are filters for the request source (name of the command-line client), request type (ADD, MOD, DEL, GET), object type that should be addressed by the request (e.g. JOB, CLUSTER_QUEUE, JOB_CLASS, ...), users that triggered the request (username) and the hostname of the host where the request is coming from. For each part of such a filter expression it is allowed to specify "*" so that the corresponding part of that expression will match any incoming request.

The full syntax for this parameter is as follows:

```
gdi_request_limits ::=
"NONE" | limit_rule [ "," limit_rule ]* .
limit_rule ::=
source ":" request_type ":" obj_type ":" user
"=" max_requests .
source ::=
"*" | "drmaa" | "qacct" | "qalter" | "qsub" | "qsh" | "qlogin" |
```

```
"qrsh" | "qconf" | "qdel" | "qhost" | "qmod" | "qquota" |
"qmon" | "qrdel" | "qrstat" | "qrsub" | "qselect" | "qstat" .
request_type ::=
"*" | "ADD" | "MOD" | "DEL" | "GET" .
obj_type ::=
"*" | "JOB" | "ADMIN_HOST" | "SUBMIT_HOST" | "EXEC_HOST" |
"CLUSTER_QUEUE" | "CPLX_ENTRY" | "CONFIG" | "MANAGER" |
"OPERATOR" | "PARALLEL_ENV" | "SCHED_CONFIG" | "USER" |
"USER_SET" | "PROJECT" | "SHARETREE_NODE" | "CKPT_ENV" |
"CALENDAR" | "HOST_GROUP" | "RESOURCE_QUOTA" |
"ADVANCE_RESERVATION" | "RESOURCE_RESERVATION" | "JOB_CLASS" |
"SESSION" | "CLUSTER" | "LICENSE_MANAGER" | "EVENT_CLIENT".
user ::= "*" | <<user_name>> .
hostname ::= "*" | <<hostname>> .
max_requests ::= <<value_>=_1>> .
```

If multiple limit rules are defined, all of them are taken into account, i.e. none of the maximum values defined in those rules is allowed to be exceeded. Requests that are not accepted will be rejected with an error message that shows the first limit rule that rejected the request. Limit rules will be tested in the order in which they appear.

Example:

```
qsub:ADD:JOB:peter:*=400,qstat:GET:JOB:*:*=400,
qstat:GET:JOB:*:poipu=10
```

The example above will limit the number of job submissions done via qsub(1) for the user named peter to a maximum of 400 submits per second.

The second and third limit rules limit the number of qstat job-get requests for all users on all hosts to 400 and to 10 for requests that are received from the host poipu. This means qstat(1) commands that will show job-related information (such as **qstat -f, qstat -j, qstat -ext,** ...) might be rejected if those limits are exceeded. In addition, commands might be rejected that do not show jobs directly but that require job information to generate the output (such as **qstat -gc** -> which shows the used job slots of queues).

gdi_request_limit replaces the functionality provided by **gdi_multi_read_req**. **gdi_multi_read_req** is deprecated as of Altair Grid Engine 8.2.

reschedule_unknown

Determines whether jobs on hosts in an unknown state are rescheduled and sent to other hosts. Hosts are registered as unknown if sge_master(8) cannot establish contact with the sge_execd(8) on those hosts (see **max_unheard**). Likely reasons are a breakdown of the host or a breakdown of the network connection in between, but another reason is that sge_execd(8) may not be executing on such hosts.

In any case, Altair Grid Engine can reschedule jobs running on such hosts to another system. **reschedule_unknown** controls the time which Altair Grid Engine will wait before jobs are rescheduled after a host state becomes unknown. The time format specification is

"hh:mm:ss" (Hours:Minutes:Seconds). If the special value "00:00:00" (or 0) is set, jobs will not be rescheduled from this host.

Rescheduling is only initiated for jobs that have activated the rerun flag (see the **-r y** option of qsub(1) and the **rerun** option of sge_queue_conf(5)). Parallel jobs are only rescheduled if the host on which their master task executes is in an unknown state. The behavior of **reschedule_unknown** for parallel jobs and for jobs without the rerun flag set can be adjusted using the **qmaster_params** settings **ENABLE_RESCHEDULE_KILL** and **ENABLE_RESCHEDULE_SLAVE.**

Checkpointing jobs will only be rescheduled when the **when** option of the corresponding checkpointing environment contains an appropriate flag. (see sge_checkpoint(5)). Interactive jobs (see qsh(1), qrsh(1)) are not rescheduled.

The default for **reschedule_unknown** is 00:00:00

The global configuration entry for this value may be overwritten by the execution host local configuration.

max_unheard

If sge_qmaster(8) could not contact or was not contacted by the execution daemon of a host for **max_unheard** seconds, all queues residing on that particular host are set to status unknown. sge_qmaster(8), at least, should be contacted by the execution daemons in order to get the load reports. Thus, **max_unheard** should be greater than the **load_report_time** (see above). The time format specification is "hh:mm:ss" (Hours:Minutes:Seconds) or just a positive integer which is interpreted as a value in seconds. Changing **max_unheard** takes immediate effect. The default for **max_unheard** is 5 minutes ("00:05:00"). This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

loglevel

This parameter specifies the level of detail that Altair Grid Engine components such as sge_qmaster(8) or sge_execd(8) use to produce informative, warning, or error messages, which are logged to the messages files in the master and execution daemon spool directories (see the description of the **execd_spool_dir** parameter above). The following message levels are available:

Value	Description
log_err	All error events being recognized are logged.
log_warning	All error events being recognized and all detected signs of potentially erroneous behavior are logged.
log_info	All error events being recognized, all detected signs of potentially erroneous behavior and a variety of informative messages are logged.

Changing **loglevel** will take immediate effect.

The default for **loglevel** is log_warning.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

max_aj_instances

This parameter defines the maximum number of array tasks to be scheduled to run simultaneously per array job. An instance of an array task will be created within the master daemon when it gets a start order from the scheduler. The instance will be destroyed when the array task finishes. Thus the parameter provides control mainly over the memory consumption by array jobs in the master and scheduler daemon. It is most useful for very large clusters and very large array jobs. The default for this parameter is 2000. The value 0 will deactivate this limit and will allow the scheduler to start as many array job tasks as can be run on suitable resources available in the cluster.

Changing **max_aj_instances** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

max_aj_tasks

This parameter defines the maximum number of array job tasks within an array job. sge_qmaster(8) will reject all array job submissions which request more than **max_aj_tasks** array job tasks. The default for this parameter is 75000. The value 0 will deactivate this limit.

Changing **max_aj_tasks** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

max_u_jobs

The number of active (not finished) jobs which each Altair Grid Engine user can have in the system simultaneously is controlled by this parameter. A value greater than 0 defines the limit. The default value 0 means “unlimited”. If the **max_u_jobs** limit is exceeded by a job submission, the submission command exits with exit status 25 and an appropriate error message.

Changing **max_u_jobs** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

max_jobs

The number of active (not finished) jobs simultaneously allowed in Altair Grid Engine is controlled by this parameter. A value greater than 0 defines the limit. The default value 0 means “unlimited”. If the **max_jobs** limit is exceeded by a job submission, the submission command exits with exit status 25 and an appropriate error message.

Changing **max_jobs** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

max_advance_reservations

The number of active (not finished) Advance Reservations simultaneously allowed in Altair Grid Engine is controlled by this parameter. A value greater than 0 defines the limit. The default value 0 means “unlimited”. If the **max_advance_reservations** limit is exceeded by an Advance Reservation request, the submission command exits with exit status 25 and an appropriate error message.

Changing **max_advance_reservations** will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

enforce_project

If set to true, users are required to request a project when submitting each job. See the **-P** option to qsub(1) for details.

Changing **enforce_project** will take immediate effect. The default for **enforce_project** is false.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

enforce_jc

If set to *true*, users are required to specify a job class when submitting each job. Default value for this parameter is false. Manager can define a default job class with the **default_jc** parameter of this configuration. This allows defining a fallback job class that will be automatically used if the user does not specify a job class.

default_jc

This parameter allows specification of a job class that will be used as the default for any job submission where the submitter does not request any job class. Default for this parameter is NONE.

enforce_user

If set to true, a sge_user(5) must exist to allow for job submission. Jobs are rejected if no corresponding user object representing the UNIX user exists.

If set to auto, a sge_user(5) object for the submitting user will automatically be created during job submission, if one does not already exist. The **auto_user_oticket**, **auto_user_fshare**, **auto_user_default_project**, and **auto_user_delete_time** configuration parameters will be used as default attributes of the new sge_user(5) object.

Changing **enforce_user** will take immediate effect. The default for **enforce_user** is auto.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

auto_user_oticket

The number of override tickets to assign to automatically created sge_user(5) objects. User objects are created automatically if the **enforce_user** attribute is set to auto.

Changing **auto_user_oticket** will affect any newly-created user objects, but will not change user objects created in the past.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

auto_user_fshare

The number of functional shares to assign to automatically-created sge_user(5) objects. User objects are created automatically if the **enforce_user** attribute is set to auto.

Changing **auto_user_fshare** will affect any newly created user objects, but will not change user objects created in the past.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

auto_user_default_project

The default project to assign to automatically-created sge_user(5) objects. User objects are created automatically if the **enforce_user** attribute is set to auto.

Changing **auto_user_default_project** will affect any newly-created user objects, but will not change user objects created in the past.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

auto_user_delete_time

The number of seconds of inactivity after which automatically-created sge_user(5) objects will be deleted. User objects are created automatically if the **enforce_user** attribute is set to auto. If the user has no active or pending jobs for the specified amount of time, the object will automatically be deleted. A value of 0 can be used to indicate that the automatically created user object is permanent and should not be automatically deleted.

Changing **auto_user_delete_time** will affect the deletion time for all users with active jobs.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

set_token_cmd

Note: Deprecated; may be removed in future release.
This parameter is only present if your Altair Grid Engine system is licensed to support AFS.

Set_token_cmd points to a command which sets and extends AFS tokens for Altair Grid Engine jobs. In the standard Altair Grid Engine AFS distribution, it is supplied as a script which expects two command-line parameters. It reads the token from STDIN, extends the token's expiration time and sets the token:

```
<set_token_cmd> <user> <token_extend_after_seconds>
```

As a shell script this command will call the programs:

- SetToken
- forge

which are provided by your distributor as source code. The script looks like the following:

```
-------------------------------
#!/bin/sh
# set_token_cmd
forge -u $1 -t $2 | SetToken
-------------------------------
```

Since it is necessary for forge to read the secret AFS server key, a site might wish to replace the **set_token_cmd** script with a command that connects to a custom daemon at the AFS server. The token must be forged at the AFS server and returned to the local machine, where SetToken is executed.

Changing **set_token_cmd** will take immediate effect. The default for **set_token_cmd** is none.

The global configuration entry for this value may be overwritten by the execution host local configuration.

pag_cmd

Note: Deprecated; may be removed in a future release.
This parameter is only present if your Altair Grid Engine system is licensed to support AFS.

The path to your pagsh is specified via this parameter. The sge_shepherd(8) process and the job run in a pagsh. Please ask your AFS administrator for details.

Changing **pag_cmd** will take immediate effect. The default for **pag_cmd** is none.

The global configuration entry for this value may be overwritten by the execution host local configuration.

token_extend_time

Note: Deprecated; may be removed in a future release.
This parameter is only present if your Altair Grid Engine system is licensed to support AFS.

The **token_extend_time** is the time period for which AFS tokens are periodically extended. Altair Grid Engine will call the token extension 30 minutes before the tokens expire until jobs have finished and the corresponding tokens are no longer required.

Changing **token_extend_time** will take immediate effect. The default for **token_extend_time** is 24:0:0, i.e. 24 hours.

The global configuration entry for this value may be overwritten by the execution host local configuration.

shepherd_cmd

Alternative path to the **shepherd_cmd** binary. Typically used to call the shepherd binary via a wrapper script or command.

Changing **shepherd_cmd** will take immediate effect. The default for **shepherd_cmd** is none.

The global configuration entry for this value may be overwritten by the execution host local configuration.

gid_range

The **gid_range** is a comma-separated list of range expressions of the form n-m (n and m are integer numbers greater than 99), where m is an abbreviation for m-m. These numbers are used in sge_execd(8) to identify processes belonging to the same job.

Each sge_execd(8) may use a separate set of group IDs for this purpose. All numbers in the group ID range have to be unused supplementary group IDs on the system, where the sge_execd(8) is started.

Changing **gid_range** will take immediate effect. There is no default for **gid_range**. The administrator will have to assign a value for **gid_range** during installation of Altair Grid Engine.

The global configuration entry for this value may be overwritten by the execution host local configuration.

qmaster_params

A list of additional parameters can be passed to the Altair Grid Engine qmaster. The following values are recognized:

AGE_LICENSE_PATH Sets the path to a valid Altair License Server, where feature licenses for Altair Grid Engine are available to be consumed. Its format is *port@hostname*, where port is the port number on which the licensing server is listening, and hostname is the name of the host on which the licensing server is running.

ALLOW_ANY_SUBMITHOSTS If this parameter is set the sge_qmaster will skip all submit host verifications. This means all incoming requests that require that the request was started at a submit host will be valid for any host. Setting this parameter to true will e.g. enable that any host can submit jobs or do qstat commands in the Altair Grid Engine cluster. It is recommended to use the default setting "false". The parameter should only be set to true if the cluster is running in a trusted environment (inhouse cluster).

ALLOW_INCREASE_POSIX_PRIORITY If this parameter is set, the POSIX priority of jobs might be increased by users up to level 0 for their own jobs even if they do not have the operator or manager role. When this parameter is absent, users are only allowed to decrease the priority of their jobs but operators and managers can increase/decrease the priority of jobs independent of their ownership.

ALLOW_REQUEST_CHANGE_FOR_ALL_USERS If this parameter is set, all users are allowed to change the assigned resources of running jobs (see qalter -when NOW), which is also the default in the absence of this parameter. It can be set to 0 to disallow the modification for all users that do not have the manager role. This parameter does not restrict modification of resource requests that will become active on reschedule (see qalter -when ON_RESCHEDULE).

ALLOW_JC_AS_VIOLATION If this parameter is set, managers are allowed to change job attributes of jobs derived from a job class where the access specifier would normally not allow adjustment.

ALLOW_PREEMPT_OWN_JOBS If this parameter is set, users are allowed to trigger manual preemption requests for their own jobs. By default only managers and operators are allowed to trigger manual preemption requests.

CL_WP_THREADS Defines the number of additional threads that are used in the sge_qmaster daemon for handling communication. This option will overwrite the value for "cl_threads" if defined in the bootstrap file line "communication_params" (See also *sge_bootstrap*(5) man page). Allowed values are the keyword "auto" (e.g. cl_wp_threads=auto), a single number (e.g. cl_wp_threads=4) or a range specification (e.g. cl_wp_threads=0-32). If this parameter is not set and there is also no "wp_threads" parameter set in the bootstrap file, the default value will be "auto". It is not supported to use more than 32 work pool threads.

Note: The sum of threads used within the sge_qmaster daemon should not be larger than the number of available CPUs for best performance. Please also check the values defined for other threads in the '$SGE_ROOT/$SGE_CELL/common/bootstrap' file.

CONSIDER_LOAD_DURING_VERIFY Changes the default behavior of qsub/qlogin/qsh/qrsh_ **-w e|w|v**. By default job validations consider only the maximum capacity of a resource on the global, host, and/or queue level, and ignore load values reported by load sensors on execution hosts. This requires managers to define the maximum capacity in the **complex_values** of the corresponding object so that job validation can be successful. This setup is recommended and therefore the default for this parameter is **false** if **CONSIDER_LOAD_DURING_VERIFY** is omitted.

Users who cannot specify the maximum capacity on global/host and/or queue level might change the behavior so that load values are not ignored by setting **CONSIDER_LOAD_DURING_VERIFY** to true. If it is set, validation for jobs might succeed even without the definition of a maximum capacity, but it will also fail if the requested amount of resources exceeds the available amount reported as load value at the current point in time.

Independent of the **CONSIDER_LOAD_DURING_VERIFY** setting, the validation process will always use the maximum capacity of a resource if it is defined and in addition a load value for this resource is reported.

DLOCK_TIMEOUT If the bootstrap parameter "debug_params" contains a valid setting for "dlock" (see also *sge_diagnostics*(5) man page), this param can be configured to specify the time interval after which a lock will be detected as a deadlock. The specified value unit is seconds. The value can be set between 1 and 1800 seconds. The default is 540 seconds which is also used if the value is not set at all. More information about deadlock detection can be found in the *sge_diagnostics*(5) man page.

DLOCK_ABORT If the bootstrap parameter "debug_params" contains a valid setting for "dlock" (see *sge_diagnostics*(5) man page) the deadlock detection thread is enabled. If a deadlock is detected, this parameter is used to force an abort() call in order to write a core file. The value can be set to 1 (true) or 0 (false). The default for standard releases is not to terminate the sge_qmaster process (DLOCK_ABORT=false). For development builds the deadlock detection will do an abort() if DLOCK_ABORT is not defined. More information about deadlock detection can be found in the *sge_diagnostics*(5) man page.

DOCKER_RESOLVE_CUDA_ID If set to "true" or "1", the qmaster will use the parameter **cuda_id** or **uuid** of an RSMAP ID (see sge_resource_map(5)) to resolve a Docker placeholder. If both parameters are configured for an RSMAP ID, **uuid** will be used. If an RSMAP ID has no **cuda_id** and no **uuid** configured or DOCKER_RESOLVE_CUDA_ID is set to "false", the qmaster will use the RSMAP ID itself to resolve the placeholder. The default value is "false".

ENABLE_ENFORCE_MASTER_LIMIT If this parameter is set, the **s_rt** and **h_rt** limits of a running job are tested and acted on by the sge_qmaster(8) when the sge_execd(8) where the job is running is in an unknown state.

After the **s_rt** or **h_rt** limit of a job is expired, the master daemon will wait additional time defined by **DURATION_OFFSET** (see sge_sched_conf(5)). If the execution daemon still cannot be contacted when this additional time is elapsed, the master daemon will force the deletion of the job (see **-f** of qdel(1)).

For jobs which will be deleted that way an accounting record will be created. For usage, the record will contain the last reported online usage when the execution daemon could contact qmaster. The **failed** state in the record will be set to 37 to indicate that the job was terminated due to a limit enforcement by the master daemon.

After the restart of sge_qmaster(8), limit enforcement begins after twice the amount of time in the largest **load_report_time** interval defined in sge_sge_conf(5) has elapsed. This will give the execution daemons enough time to reregister with the master daemon.

ENABLE_FORCED_QDEL_IF_UNKNOWN If this parameter is set, a deletion request for a job is automatically interpreted as a forced deletion request (see **-f** of qdel(1)) if the host where the job is running is in unknown state.

ENABLE_JOB_VERIFY_BEFORE_JSV Enables additional job validation of incoming jobs to be done before corresponding job parameters are passed to JSV. Find more information concerning job validation in qsub(1) -w and concerning JSV in sge_jsv(5).

ENABLE_RESOURCE_UTILIZATION_TRACING Allowed settings are NONE, ERROR, WARNING and INFO. If this parameter is set to a value != NONE the qmaster daemon will do additional verification of resource bookings. The qmaster daemon will log information for requested job resources into the messages file. If set to ERROR only critical booking errors will be printed to the messages file. The WARNING value will also show warning messages when an available resource gets negative, which can happen when the complex capacity is reduced while running jobs still hold the bookings. The level INFO will show all kinds of bookings. Default is NONE which means tracing is disabled. The resource tracing is currently only available at host level.

ENABLE_RESOURCE_UTILIZATION_TRACING_COMPONENT This optional parameter accepts the values MASTER, SCHEDULER or ALL. This optional parameter can be used to specify for which component inside qmaster the tracing should be done. Default is MASTER which shows bookings for all threads in qmaster, but not the scheduler thread. The SCHEDULER setting will only show tracing for the scheduler thread. The setting ALL will show bookings for all qmaster threads.

ENABLE_SUP_GRP_EVAL By default all UNIX group entries in access lists, including those in manager or operator lists, will only be evaluated against the primary UNIX group of each user. If such group entries should also be evaluated against their secondary groups, this paramter can be defined.

ENABLE_UPDATE_CONFIG_NOTIFICATION THIS PARAMETER IS NOT SUPPORTED AND MAINLY USED FOR TESTING PURPOSES! By default the execd hosts will obtain configuration changes per load report interval. With this parameter the respective execution daemons will be triggered to request their configuration immediately after a configuration change. If this parameter is enabled (value set to “true” or “1”) there might be a performance impact on installations with a huge number of execution daemons. The default for this parameter is “false” or “0”. Changing this parameter will have immediate effect.

ENABLE_FORCED_QDEL If this parameter is set, non-administrative users can force deletion of their own jobs via the -f option of qdel(1). Without this parameter, forced deletion of jobs is only allowed by the Altair Grid Engine manager or operator.

Note: Forced deletion for jobs is executed differently depending on whether users are Altair Grid Engine administrators or not. For administrative users, the jobs are removed from the internal database of Altair Grid Engine immediately. For regular users, the equivalent of a normal qdel(1) is executed first, and deletion is forced only if the normal cancellation was unsuccessful.

FORBID_RESCHEDULE If this parameter is set, re-queuing of jobs will not be initiated, whether or not the return value of any job script is set to 99. Without this parameter, jobs returning

the value 99 are rescheduled. This can be used to cause the job to be restarted at a different machine, for instance if there are not enough resources on the current one.

FORBID_APPERROR If this parameter is set, the job application cannot set itself to an error state. Without this parameter, jobs returning the value 100 are set to an error state (and therefore can be manually rescheduled by clearing the error state). This can be used to set the job to an error state when a starting condition of the application is not fulfilled before the application itself has been started, or when a cleanup procedure (e.g. in the epilog) decides that it is necessary to run the job again, by returning 100 in the prolog, pe_start, job script, pe_stop, or epilog script.

DISABLE_AUTO_RESCHEDULING **Note:** Deprecated; may be removed in a future release.
If set to "true" or "1", the reschedule_unknown parameter is not taken into account.

DISABLE_HOST_ALIASES_CHECK If set to "true" or "1", the *sge_qmaster*(8) will stop checking the status of the host_aliases file. The default behaviour is that qmaster checks the status of the host_aliases file every 60 seconds. If set to "true" the host_aliases file (<sge_root>/<cell>/common/host_aliases) will only be re-read by qmaster if the *qconf -uha* option is used (see *qconf*(1) man page).

DISABLE_QINSTANCE_CHECK If set to "true" or "1", a job verification process is disabled that checks that at least one qinstance is available for a submitted job that can be accessed by it. It will also disable the verification that the specified queue must exist when a job is added or modified. If the check is enabled and if no qinstance is available for a job, that job will be rejected. In the absence of this parameter, this check is enabled.

ENABLE_BINDING_RECORDS If set to "true" or "1", the reporting for granted binding (55), granted rsmap (56), granted devices (57) and granted req. (73) will be enabled. (The records are described in sge_accounting(5) and sge_reporting(5)). The default for this parameter is "false" or "0" which means the reporting is turned off.

ENABLE_RESCHEDULE_KILL If set to "true" or "1", the *reschedule_unknown* parameter affects jobs which do not have the rerun flag activated (see the **-r y** option of qsub(1) and the **rerun** option of sge_queue_conf(5)); these jobs are simply finished as they can't be rescheduled.

ENABLE_RESCHEDULE_SLAVE If set to "true" or "1", Altair Grid Engine additionally triggers job rescheduling when the host where the slave tasks of a parallel job execute is in an unknown state, if the reschedule_unknown parameter is activated.

ENABLE_XD_RUN_AS_IMAGE_USER If set to "true" or "1", the submit option *-xd_run_as_image_user* can be used to let autostart Docker jobs be started as the user defined in the Docker image, not as the job user. The default is "false".

LICENSE_TYPE If set to "GRID_ENGINE_FEATURE", or unset, legacy GridEngine feature licenses will be used from the server or file specified in the AGE_LICENSE_PATH qmaster_params value. If set to "ALTAIR_UNITS", new Altair-Units HyperWorks licenses will be used with the server specified in AGE_LICENSE_PATH.

LICENSING_MAIL In order to determine the recipients of licensing related emails, LICENSING_MAIL has to be set. Its value has to be a valid email address or a list of email addresses separated by a colon. If the parameter is not set, the administrator_mail configuration parameter is used instead if provided.

MAX_DYN_EC Sets the max number of dynamic event clients (as used by qsub -sync y and by Altair Grid Engine DRMAA API library sessions). The default is set to 1000. The number

of dynamic event clients should not be larger than half of the number of file descriptors the system has. The file descriptors are shared among the connections to all exec hosts, all event clients, and file handles that the qmaster needs.

MAX_AJ_QFAIL_HOLD This parameter can be used to set a hold state on all queued tasks of a job if the job is responsible for setting a queue to an error state. The value set for this parameter defines how many queue errors a job can produce without being set to a hold state. Jobs that produce a queue error are usually rescheduled to another queue. This parameter can be used to prevent the situation wherein a job is responsible for setting all or many queues into an error state. The default value for this paramter is 0 (zero) which means that no job will be set automatically to a hold state. Any other positive number is used as a threshold value and jobs and their tasks will be set to a user hold when the queue error counter reaches the configured value. The “qstat -explain E” and “qstat -j ” command will show when a job was set to a hold state via this option.

MAX_AJ_TFAIL_HOLD This parameter can be used to set a hold state on all queued tasks if one or more tasks report a job error. The value set for this parameter defines how many task errors a job can produce without being set to a hold state. This parameter can be used to prevent the situation that a job might produce lots of needless spooling overhead at the sge_qmaster daemon. The default value for this paramter is 0 (zero) which means that no job will be set automatically into a hold state. Any other positive number is used as threshold value and jobs and their tasks will be set to a user hold when the task error counter reaches the configured value. The “qstat -j ” command will show when a job was set to a hold state via this option.

MONITOR_TIME Specifies the time interval when monitoring information should be printed. Monitoring is disabled by default and can be enabled by specifying an interval. Monitoring is per thread and is written to the messages file or displayed via the “qping -f” command-line tool. Example: MONITOR_TIME=0:0:10 generates and prints the monitoring information approximately every 10 seconds. The specified time is a guideline only and not a fixed interval. The interval that is actually used is printed. In this example, the interval could be anything between 9 seconds and 20 seconds. Additional information can be obtained if, in addition, profiling output is enabled (see PROF_WORKER or PROF_EXECD parameters in execd_params).

MONITOR_REQUEST_QUEUES If set to “true” or “1”, additional information about the qmaster internal request queues will be provided in the monitoring output of qping. Find more information in sge_diagnostics(5).

LOG_MONITOR_MESSAGE Monitoring information is logged in the messages files by default. This information can be accessed via *qping*(1). If monitoring is always enabled, the messages files can become quite large. This switch disables logging in the messages files, making qping -f the only source of monitoring data.

MONITOR_JEMALLOC If this parameter is set, statistics and configuration data of the jemalloc memory allocator are dumped (appended) to the file jemalloc.dump in the qmaster spool directory in every monitoring interval (see *MONITOR_TIME*).

The jemalloc memory allocator is used on the Altair Grid Engine architecture *lx-amd64*.

Setting *MONITOR_JEMALLOC* to an empty string (specifying *MONITOR_JEMALLOC*=) will dump information using default settings. Output format and contents can be influenced by specifying the following characters for the value of *MONITOR_JEMALLOC*:

character	description
J	output the statistics in JSON format instead of human-readable format
a	omit per-arena statistics
b / l	omit per-size bin and large objects
e	omit extent statistics
x	omit mutex statistics

For example *MONITOR_JEMALLOC=Jx* will output statistics data in JSON format but will not output mutex statistics.

Please note that a significant amount of data is generated by this setting and the jemalloc.dump will grow rapidly. Only enable the parameter when asked by a Altair Grid Engine support engineer and for a short time.

See also jemalloc documentation in $SGE_ROOT/doc/jemalloc/jemalloc.html.

PROF_SIGNAL Enables the profiling for qmaster signal thread (e.g. PROF_SIGNAL=true). More information about profiling can be found at the PROF_WORKER parameter description.

PROF_WORKER Enables the profiling for qmaster worker threads (e.g. PROF_WORKER=true).

Profiling provides the user with the ability to get system measurements. This can be useful for debugging or optimization of the system. The profiling output will be printed to the messages file. The output interval can be set with the parameter PROF_TIME to be in sync with the monitoring output.

PROF_READER Enables profiling for qmaster reader threads (e.g. PROF_READER=true). More information about profiling can be found at PROF_WORKER parameter description.

PROF_LISTENER Enables profiling for qmaster listener threads (e.g. PROF_LISTENER=true). More information about profiling can be found at PROF_WORKER parameter description.

PROF_DELIVER Enables profiling for the qmaster event deliver thread (e.g. PROF_DELIVER=true). More information about profiling can be found at PROF_WORKER parameter description.

PROF_API Enables profiling for the qmaster api thread (e.g. PROF_API=true). More information about profiling can be found at PROF_WORKER parameter description.

PROF_STREAMING Enables profiling for the qmaster streaming thread (e.g. PROF_STREAMING=true). More information about profiling can be found at PROF_WORKER parameter description.

PROF_TEVENT Enables profiling for the qmaster timed event thread (e.g. PROF_TEVENT=true). More information about profiling can be found at PROF_WORKER parameter description.

PROF_TIME Specifies the time interval when profiling information should be printed for the enabled PROF_xxxx threads in the sge_qmaster daemon (e.g. PROF_WORKER=true). If the value is set to 00:00:00 (default) the profiling output will be printed synchronized with the monitor output. If monitoring is also disabled (see MONITOR_TIME parameter) the default output interval is set to 00:01:00 (60 seconds). Any other timeout value will print out the profiling information at the specified time.

PROF_JOB_DELETION This parameter only works together with PROF_WORKER=true. With PROF_JOB_DELETION=true there will be no interval based profiling output for the worker thread anymore. Profiling output will be printed when a worker thread runs through the

job deletion code. That profiling information then relates exclusively to the last delete job action that the corresponding thread did.

PROF_JOB_ADD This parameter only works together with PROF_WORKER=true. With PROF_JOB_ADD=true there will be no interval based profiling output for the worker thread anymore. Profiling output will be printed when a worker thread adds a new job to the system. The profiling information then relates exclusively to the last job submit action that was handled by the corresponding thread.

PROF_COMMLIB_TIME Enables the profiling for the communication library. The value specifies the log interval for commlib profiling to be printed in the messages file. The logging shows the number of connected clients, the number of buffered messages at commlib layer (incoming/outgoing), the memory needed in the commlib layer for the buffered messages (incoming/outgoing), the number of cached resolved hostnames and the number of currently active commlib work pool threads.

LOG_INCOMING_MESSAGE_SIZE This parameter is used to specify whether profiling information about incoming requests is logged in the messages file. The specified value will be used as a threshold. All incoming messages needing more memory size than specified will be logged. Default value for this parameter is 0 which means the feature is turned off. (e.g. LOG_INCOMING_MESSAGE_SIZE=20M)

LOG_OUTGOING_MESSAGE_SIZE This parameter is used to specify whether profiling information about outgoing requests is logged in the messages file. The specified value will be used as threshold. All outgoing messages needing more memory size than specified will be logged. Default value for this parameter is 0 which means the feature is turned off. (e.g. LOG_OUTGOING_MESSAGE_SIZE=20M)

MAX_INCOMING_MESSAGE_SIZE This parameter is used to specify a message size limit for accepting incoming requests. All incoming client requests using more memory than specified are rejected. The client will get an error message for the request. The value cannot be set below 1M. Values < 1M will be interpreted as 0 (=turned off). All rejected client requests are logged in the messages file. Default value for this parameter is 0 which means the feature is turned off. (e.g. MAX_INCOMING_MESSAGE_SIZE=1G)

MAX_OUTGOING_MESSAGE_SIZE This parameter is used to specify a message size limit for creating client responses such as qstat -j "*". All client requests that result in creating a response message exceeding the specified memory size will get an error message. The value cannot be set below 1M. Values < 1M will be interpreted as 0 (=turned off). All rejected client requests are logged in the messages file. Default value for this parameter is 0 which means the feature is turned off. (e.g. MAX_OUTGOING_MESSAGE_SIZE=1G)

STREE_SPOOL_INTERVAL Sets the time interval for spooling the sharetree usage. The default is set to 00:04:00. The setting accepts a colon-separated string or seconds. There is no setting to turn the sharetree spooling off. (e.g. STREE_SPOOL_INTERVAL=00:02:00)

MAX_JOB_DELETION_TIME Sets an upper bound on how long the qmaster will spend deleting jobs. After this time, the qmaster will continue with other tasks and schedule the deletion of remaining jobs at a later time. The default value is 3 seconds, and will be used if no value is entered. The range of valid values is > 0 and <= 5. (e.g. MAX_JOB_DELETION_TIME=1)

MAX_MASTER_TASK_WAIT_TIME Sets an upper bound on how long the qmaster will wait to get all slave task reports for parallel jobs when the master task is already finished. The value is

the waiting time in seconds. The range of valid values is >= 20 and <= 720. The default for this parameter is to wait 20 seconds. (e.g. MAX_MASTER_TASK_WAIT_TIME=30)

ENABLE_JOB_FAILURE_IF_SLAVE_TASK_MISSING If this parameter is set to true a missing slave task report of a tightly integrated parallel job will set the failed state of the master task to 101. If the master task is already in a failure state the value of the master task will not be overwritten. (e.g. ENABLE_JOB_FAILURE_IF_SLAVE_TASK_MISSING=true)

ENABLE_JOB_FAILURE_ON_SLAVE_TASK_ERROR If this parameter is set to true a slave task which reports a failure or reports a non-zero exit status will automatically set the failed state for the master task of the parallel job. The first slave job which reports a non-zero exit status will set the master task failure field in the accounting file to the value 102. If a slave task reports some general failure the master task failure state will be set to 103. If the master task is already in a failure state the value will not be overwritten unless a slave task was killed by the execution daemon because it reached an active limit. In this case the master task will get the failure state of the corresponding slave task. This option is only valid for tightly integrated jobs. (e.g. ENABLE_JOB_FAILURE_ON_SLAVE_TASK_ERROR=true)

LOST_JOB_TIMEOUT If this timeout parameter is set, the qmaster worker threads will monitor the jobs reported by the execution daemons. If a task of a job that was started on an execution node is not reported for longer than the defined timeout the job is logged in the qmaster messages file. Job loss is e.g. possible if an execution daemon cannot read one or more files in its spooling directory at startup. This can happen when the spooling directory runs out of disc space or for any other possible file problems. Such jobs typically are shown as running and occupy a slot on the execution daemon indefinitely (see also "enable_lost_job_reschedule"). If an execution daemon is not online or came online shortly the timeout will be extended until all preconditions are fulfilled. The minimum timeout also depends on the **max_unheard** and **load_report_time** settings. If the timeout is set below the allowed minimum timeout, the calculated minimum timeout is used. The resulting timeout will be logged in the qmaster messages file. If the parameter is changed the job timeouts will be reinitialized. If the timeout is set to 00:00:00 the lost job detection is turned off. This is also the default setting for this parameter. The timeout is specified in seconds.

ENABLE_LOST_JOB_RESCHEDULE This parameter is only valid if there is a "lost_job_timeout" parameter configured. If it is enabled the jobs for which the timeout was detected are set to an error state and will show up again in the pending job list. The accounting record will contain the "failed" state 22. Such jobs will not occupy a slot on the execution node and the slots are free again for other jobs. The administrator might remove the error state and let the job run again or just delete them after solving the reported problem. The default for "enable_lost_job_reschedule" is false.

GDI_TIMEOUT Sets how long the communication library will wait for gdi send/receive operations. The default value is set to 60 seconds. After this time, the communication library will retry, if "gdi_retries" is configured, receiving the gdi request. When "gdi_retries" is not configured, the communication library will return with a "gdi receive failure" (e.g. gdi_timeout=120 will set the timeout time to 120 sec). The default value for gdi_timeout is 60 seconds.

GDI_REQUEST_SESSION_TIMEOUT Default duration of a session as defined for "time" in sge_types(1). When this value is not defined, 00:15:00 (= 900 seconds) will be used by default duration for new sessions. Changing this value will not change the duration of existing sessions.

MAX_READER_DELAY If defined, the value for this parameter has to be an integer value in the

range from 0 to 5000. It defines the number of milliseconds before the event processing thread of the read-only thread thread-pool will enforce the update of the read-only data store.

0 means that the event processing thread will interrupt all other read-only threads as soon as possible (regularly when currently processed requests are finished) so that it can update the read-only thread data store immediately.

With values >0 the event processing thread also tries to process immediately but if there are pending read-only requests, handling of these requests will be preferred as long as the defined time value has not elapsed.

When this value is not specified, the reader delay value used is 1000, which is also the recommended value for up to 8 read-only threads. If more read-only threads are started, it is recommended to increase the delay (8-16 threads => 2500; 16-32 threads => 3750; 32-64 threads 5000).

Please note that the delay is the same that you might see for command-line clients that use a session (see sge_session_conf(5))

ENFORCE_GDI_WORKER GDI (Grid Engine Data Interface) is the name of an interface that command-line clients use to communicate with qmaster.

When enforce_gdi_worker is set to 1, all GDI requests (read-only and read-write) will be handled by worker threads in sge_qmaster(8) even if reader threads are activated. Requests will then be handled in an FCFS manner as it was done in prior versions of Altair Grid Engine.

Read-only threads can also be disabled by setting the bootstrap parameter reader to 0. This not only disables reader threads but also disables the creation of the read-only thread pool in sge_qmaster(8). Please note that changing the bootstrap file requires restarting sge_qmaster(8) before the changes take effect.

ENFORCE_GDI_READER_FOR_EXECD If set to “true” or “1”, incoming get requests from the execution daemons are handled by the reader threads if such threads are configured in the bootstrap configuration file. Default value for this parameter is true.

GDI_RETRIES Sets how often the gdi receive call will be repeated until the gdi receive error appears. The default is set to 1. In this case the call will be done 1 time with no retry. When the value is -1 the call will be done permanently. In combination with the gdi_timeout parameter it is possible to configure a system with eg. slow NFS to make sure that all jobs will be submitted. (e.g. gdi_retries=4)

CL_PING Turns on/off a communication library ping. This parameter will create additional debug output. This output shows information about the error messages which are returned by communication and it gives information about the application status of the qmaster. eg, if it is unclear what is the reason for gdi timeouts, this may show you some useful messages. The default value is false (off) (e.g cl_ping=false)

SCHEDULER_TIMEOUT Setting this parameter allows the scheduler GDI event acknowledge timeout to be manually configured to a specific value. Currently the default value is 10 minutes with the default scheduler configuration and limited between 600 and 1200 seconds. Value is limited only in case of default value. The default value depends on the current scheduler configuration. The SCHEDULER_TIMEOUT value is specified in seconds.

JSV_TIMEOUT This parameter measures the response time of the server JSV. In the event that the response time of the JSV is longer than the timeout value specified, this will cause the JSV

to be re-started. The default value for the timeout is 10 seconds. If the value is modified, it must be greater than 0. When the timeout is reached, the JSV will tries to re-start only once; if the timeout is reached again an error will occur.

JSV_THRESHOLD The threshold of a JSV is measured as the time it takes to perform a server job verification. If this value is greater than the user-defined value, it will cause logging to appear in the qmaster messages file at the INFO level. By setting this value to 0, all jobs will be logged in the qmaster messages file. This value is specified in milliseconds and has a default value of 5000.

GDI_THRESHOLD When processing a gdi request (e.g. submitting a job or querying job information via qstat) takes too long, a warning is printed in the qmaster messages file. The time being considered too long can be defined by setting gdi_threshold in seconds. Default is a threshold of 60 seconds.

OLD_RESCHEDULE_BEHAVIOR Beginning with version 8.0.0 of Altair Grid Engine, the scheduling behavior changed for jobs that are rescheduled by users. Rescheduled jobs will not be put at the beginning of the pending job list any more. The submit time of those jobs is set to the end time of the previous run. Because of that, those rescheduled jobs will be appended at the end of the pending job list as if a new job had been submitted. To achieve the old behavior the parameter OLD_RESCHEDULE_BEHAVIOR has to be set.

OLD_RESCHEDULE_BEHAVIOR_ARRAY_JOB Beginning with version 8.0.0 of Altair Grid Engine, the scheduling behavior changed for array job tasks that are rescheduled by users. As soon as an array job task gets scheduled, all remaining pending tasks of that job will be put at the end of the pending job list. To achieve the old scheduling behavior the parameter OLD_RESCHEDULE_BEHAVIOR_ARRAY_JOB has to be set.

ENABLE_SUBMIT_LIB_PATH Beginning with version 8.0.1p3 of Altair Grid Engine, environment variables like LD_PRELOAD, LD_LIBRARY_PATH and similar variables by default may no longer be set via submit option -v or -V.

Setting these variables could be misused to execute malicious code from user jobs, if the execution environment contained methods (e.g. prolog) to be executed as the root user, or if the old interactive job support (e.g. via ssh) was configured.

Should it be necessary to allow setting environment variables like LD_LIBRARY_PATH (except LD_PRELOAD; see ENABLE_SUBMIT_LD_PRELOAD) via submit option -v or -V, this can be enabled again by setting ENABLE_SUBMIT_LIB_PATH to TRUE.

In general the correct job environment should be set up in the job script or in a prolog, making the use of the -v or -V option for this purpose unnecessary.

ENABLE_SUBMIT_LD_PRELOAD Setting this variable could be misused to execute malicious code from user jobs, if the execution environment contained methods (e.g. prolog) to be executed as the root user, or if the old interactive job support (e.g. via ssh) was configured.

Should it be necessary to allow setting LD_PRELOAD via submit option -v or -V, this can be enabled again by setting ENABLE_SUBMIT_LD_PRELOAD to TRUE.

In general the correct job environment should be set up in the job script or in a prolog, making the use of the -v or -V option for this purpose unnecessary.

See also ENABLE_SUBMIT_LIB_PATH for more information.

ENABLE_SUBMIT_LIB_ENV_PREFIX Beginning with version 8.5.5 of Altair Grid Engine, environment variables starting with the prefix LD_, LDR_ and DYLD_ may no longer be set via submit option -v or -V.

Setting these variables could be misused to execute malicious code from user jobs, if the execution environment contained methods (e.g. prolog) to be executed as the root user, or if the old interactive job support (e.g. via ssh) was configured.

Should it be necessary to allow setting environment variables starting with one of these prefixes it is possible to enable them again by setting ENABLE_SUBMIT_LIB_ENV_PREFIX to TRUE.

In general the correct job environment should be set up in the job script or in a prolog, making the use of the -v or -V option for this purpose unnecessary.

See also ENABLE_SUBMIT_LIB_PATH and ENABLE_SUBMIT_LD_PRELOAD for more information.

ENABLE_SUBMIT_LIB_WARNING This *qmaster_params* parameter can be used to show a warning message in the submit client if at job submission potentially harmful environment variables such as LD_LIBRARY_PATH or LD_PRELOAD were exported from the submit user's environment with the "-v" or "-V" flags. Those variables can be harmful when e.g. a prolog or epilog script is executed under a different user id than the user id of the submit user.

The warning message is only printed if those variables would be removed which is the default unless through *qmaster_params* ENABLE_SUBMIT_LD_PRELOAD, ENABLE_SUBMIT_LIB_PATH, or ENABLE_SUBMIT_LIB_ENV_PREFIX these variables are not removed.

If ENABLE_SUBMIT_LIB_WARNING is set to "true" the submission client will print a warning message. The default is "false".

ALLOW_EMPTY_AFS_TOKEN This parameter is considered only if Altair Grid Engine is installed with AFS support. If this parameter is set to TRUE, the AFS token generation can be done with the **set_token_cmd** only. The configured script can be used to completely generate the token at job execution time. The default method to generate the token by setting up the script $sge_ROOT/util/get_token_cmd is still active with this setting, but it will not result in an error if the get_token_cmd script is not available. If this parameter is set to TRUE moving away the get_token_cmd script is suggested to get a better submit performance.

MAX_JOB_ID This parameter can be used for setting the maximum job ID used by Altair Grid Engine. Job IDs are allocated from ID 1 to the maximum set.

Setting MAX_JOB_ID to 0 disables job submission.

The default maximum job ID is 4294967295 (the maximum 32bit number).

This parameter also has effect on the advance reservation IDs.

MIN_PENDING_ENROLLED_TASKS Ticket calculation for sharetree, functional, and override policy is done per job and for already existing tasks of array jobs. Already existing array tasks are running array tasks (getting running tickets) and array tasks which have been running but are pending again, e.g. due to rescheduling (getting pending tickets). If different pending tickets shall be computed for tasks of an array job it is necessary to create (enroll) pending array tasks. This can be controlled via the MIN_PENDING_ENROLLED_TASKS parameter.

Default setting is 0, in which case no pending array tasks will be enrolled for ticket calculation, and the tasks are not created before they are scheduled. Pending tickets are calculated for the whole job; all tasks of an array job will get the same amount of tickets.

Setting it to a positive number triggers creation of this number of pending array tasks per job.

When it is set to -1 all tasks of an array job will get enrolled.

SGE_DEBUG_LEVEL With the environment variable SGE_DEBUG_LEVEL debug output of sge_qmaster can be enabled when sge_qmaster is running non-daemonized. The qmaster_params SGE_DEBUG_LEVEL serves the same purpose but can be switched on and off during runtime. See sge_diagnostics(5) for details about SGE_DEBUG_LEVEL. Please note that the delimiter between multiple levels is : (colon) for the qmaster_params, so the environment setting SGE_DEBUG_LEVEL="3 0 0 0 0 0 0 0" translates to qmaster_params SGE_DEBUG_LEVEL=3:0:0:0:0:0:0:0.

By default, debug output goes to stderr, which means that with a daemonized sge_qmaster it would get lost. Please use the qmaster_params SGE_DEBUG_TARGET to redirect debug output to a file.

If SGE_DEBUG_LEVEL is specified in execd_params, the debug level for the sge_execd is set.

SGE_DEBUG_TARGET With the SGE_DEBUG_TARGET parameter, debug output of sge_qmaster can be redirected into a file, to stdout, or to stderr. Default is stderr if the parameter is not set or if a given file cannot be opened.

If SGE_DEBUG_TARGET is set in execd_params, the sge_execd debug output is redirected to the file specified.

JOB_SPECIFIC_TRACING_ID If this qmaster_params attribute is set to a job ID, during the Scheduler run for this job both the schedd_runlog and debug tracing is enabled. After the job is scheduled, it is automatically disabled again. See also the qmaster_params attributes **JOB_SPECIFIC_TRACING_DEBUG_LEVEL** and **JOB_SPECIFIC_TRACING_TARGET** for meaningful debug levels and trace targets. This qmaster_params attribute causes the qmaster to overwrite any debug target or debug level specified in the environment or the **qmaster_params** and does not restore them after the specified job was scheduled!

Example: qmaster_params JOB_SPECIFIC_TRACING_ID=3000000036

JOB_SPECIFIC_TRACING_DEBUG_LEVEL If **JOB_SPECIFIC_TRACING_ID** is set to a job ID and **JOB_SPECIFIC_TRACING_DEBUG_LEVEL** is set to a debug level like it can be set with the **qmaster_params** attribute **SGE_DEBUG_LEVEL** (see sge_conf(5)), tracing to the target specified with **JOB_SPECIFIC_TRACING_TARGET** is enabled according to the specified level. This qmaster_params attribute overwrites any debug level specified in the environment or the **qmaster_params** and does not restore it after the specified job was scheduled!

Example: qmaster_params JOB_SPECIFIC_TRACING_DEBUG_LEVEL=3:0:0:0:0:0:0:0

JOB_SPECIFIC_TRACING_TARGET If **JOB_SPECIFIC_TRACING_ID** is set and tracing is enabled by setting a valid **JOB_SPECIFIC_TRACING_DEBUG_LEVEL**, the file where the trace output is written can be defined by specifying the **JOB_SPECIFIC_TRACING_TARGET**. See also **SGE_DEBUG_TARGET** in sge_conf(5). The target and the scheduler runlog are overwritten every time the scheduler starts to schedule the given job, which means they are overwritten not only with every scheduler run, but could also be overwritten within one scheduler run

if the job is an array job where different array tasks are scheduled in different steps. This qmaster_params attribute overwrites any debug target specified in the environment or the **qmaster_params** and does not restore it after the specified job was scheduled!

Example: qmaster_params JOB_SPECIFIC_TRACING_TARGET=/tmp/trace.log

AR_RESERVE_AVAILABLE_ONLY When this parameter is set to 1 or true, advance reservations submitted via qrsub will only be scheduled to currently available resources; advance reservations are not scheduled to queue instances that are disabled, suspended, in error state or unknown. This is a cluster-wide setting which when enabled will overwrite the AR specific setting done via qrsub option **-rao**; see qrsub(1).

MAX_TCON_TASKS This parameter can be used to disable (value 0) concurrent array jobs or limit the maximum size of concurrent array jobs. Submission of concurrent array jobs will be rejected if their size (number of array tasks) exceeds the value of MAX_TCON_TASKS. See also documentation of the -tcon submit option in submit(1).

MAX_AR_CAL_DEPTH Can be used for increasing or decreasing the maximum allowable calendar depth for Standing Reservation requests using **qrsub -cal_depth**. By default the limit is set to 8.

MAX_AR_CAL_JMP Can be used for increasing or decreasing the maximum allowable unallocated (skippable) Standing Reservation instances. By default the maximum is set to 8. The default request for a Standing Reservation is 0.

RESCHEDULE_AR_INTERVAL This parameter allows rescheduling of reservations for advance reservations that are inan error state (optionally, see also RESCHEDULE_AR_ON_ERROR) and for standing reservations that are either not allocated (as no resources were available when the standing reservation was scheduled) or in an error state. When RESCHEDULE_AR_INTERVAL is set to 0 (default), no rescheduling is done. A value greater 0 is the interval in seconds in which sge_qmaster tries to reschedule reservations.

RESCHEDULE_AR_TIMEOUT When the rescheduling of advance or standing reservations takes longer than the timeout specified in this parameter it is stopped. In the next RESCHEDULE_AR_INTERVAL sge_qmaster will take up rescheduling at the point where it stopped. The timeout is specified as a decimal number in seconds; default is 0.5 seconds.

RESCHEDULE_AR_ON_ERROR This parameter defines whether only standing reservations that do not have an allocation are rescheduled, or also advance or standing reservations that are in an error or warning state. Default is FALSE: Only standing reservations having no allocation are rescheduled.

AR_DETACH_JOBS_ON_RESCHEDULE When advance/standing reservations are modified via qralter(1) and the modification requires re-scheduling of jobs, re-scheduling can only be done if there are no jobs currently running in the AR/SR. When the qmaster_param AR_DETACH_JOBS_ON_RESCHEDULE is set to TRUE and re-scheduling of an AR/SR is necessary, jobs running in the AR/SR will be detached from the AR/SR which in turn allows re-scheduling of the AR/SR. Whether a job is detached from an AR/SR can be seen in the output of qstat -j job_id via the attribute ar_detached. Default for AR_DETACH_JOBS_ON_RESCHEDULE is FALSE, do not detach jobs.

DISABLE_NAME_SERVICE_LOOKUP_CACHE This parameter can be used to disable (value 1) caching of name service lookup calls. The default setting is that caching is enabled (value 0). Switching off the cache might decrease the performance significantly.

NAME_SERVICE_LOOKUP_CACHE_ENTRY_LIFE_TIME This parameter can be used to define when a cached entry in the name service lookup cache is removed. The default setting is zero (value 0). The default setting will auto-adjust the timeout for compatibility reasons to 600 seconds. The value cannot be set > 86400 (1 day). Changing this parameter might have a significant performance influence.

NAME_SERVICE_LOOKUP_CACHE_ENTRY_UPDATE_TIME This parameter can be used to define when a cached entry in the name service should get re-resolved. The default setting is zero (value 0). The default setting will auto-adjust the timeout for compatibility reasons to 120 seconds. The value cannot be set > 1800 (30 min). Changing this parameter might have a significant performance influence.

NAME_SERVICE_LOOKUP_CACHE_ENTRY_RERESOLVE_TIME This parameter can be used to define when a cached entry that was not resolvable (name service returned an error) when the host was added or at the last cache entry update. The default setting is zero (value 0). The default setting will auto adjust the timeout for compatibility reasons to 60 seconds. The value cannot be set > 600. Changing this parameter might have a significant performance influence.

LOG_JOB_VERIFICATION_TIME This parameter can be used to enable profiling logging in the qmaster messages file for the job verification with the submission option -w; see also submit(1).

When the parameter is not set or it is set to a negative value, no profiling will be done.

When the parameter is set to 0, profiling will be done and the resulting verification time will always be logged as an INFO message.

When the parameter is set to a value (threshold) > 0, profiling will be done; if the verification time exceeds the given threshold a WARNING message will be logged.

The message logging will be forced and will also be done if current loglevel is not set to INFO or WARNING.

LOG_REQUEST_PROCESSING_TIME This parameter can be used to enable profiling logging in the qmaster messages file for the processing of requests by worker, reader and event master threads.

When the parameter is not set or it is set to a negative value, no profiling will be done.

When the parameter is set to 0, profiling will be done and the resulting processing time will always be logged as an INFO message.

When the parameter is set to a value (threshold specified in seconds as decimal number) > 0, profiling will be done; if the processing time exceeds the given threshold a WARNING message will be logged.

The message logging will be forced and will also be done if current loglevel is not set to INFO or WARNING.

LOG_SPOOLING_TIME This parameter can be used to enable profiling logging in the qmaster messages file for spooling operations.

When the parameter is not set or it is set to a negative value, no profiling will be done.

When the parameter is set to 0, profiling will be done and the resulting spooling time will always be logged as an INFO message.

When the parameter is set to a value (threshold specified in seconds as decimal number) > 0, profiling will be done; if the spooling time exceeds the given threshold a WARNING message will be logged.

The message logging will be forced and will also be done if current loglevel is not set to INFO or WARNING.

LOG_RESOLVING_TIME This parameter can be used to enable time information logging in the qmaster messages file for hostname or IP address lookup operations. The parameter can be specified as a decimal number in seconds (e.g. "0.01"). When the parameter is not set or it is set to a negative value no logging will be done. When the parameter is set to 0, logging will be enabled and the measured time for any resolving lookup will be logged as an INFO message. When the parameter is set to a value > 0, the measured time will be logged only if the given threshold has been reached. The resulting logging information will be printed as a WARNING message. When the parameter is not set or it is set to a negative value only lookups lasting more than 15 seconds will be logged as a WARNING message. The message logging will be forced and will also be done if current loglevel is not set to INFO or WARNING.

REJECT_CRLF_SHEBANG This parameter can be used to reject scripts that are meant for UNIX-like operating systems but have a CR-LF line ending (\r\n or ^M). If this is the case, e.g. with **#!/bin/bash^M**, the job cannot start and will result in an error state. The reason for the error is that the interpreter cannot be found, hence leading to the error-message "No such file or directory".

If this parameter is set to **true**, such a job script will be rejected on submission.

SUBMIT_JOBS_WITH_AR_NAME This parameter can be used to enable job submissions that have advance reservation names (along with the ID's in the existing -ar switch). This will also ensure that advance reservations are created with unique names. For this parameter to be set to **true**, all existing advance reservations should have a unique name, if they were already assigned one.

Default value for this parameter is set to **false**, which allows jobs to be submitted with only advance reservation ID's in the -ar switch. This also allows for non-unique names to be assigned for advance reservations.

Changing **qmaster_params** will take immediate effect, except for gdi_timeout, gdi_retries, and cl_ping; these will take effect only for new connections. The default for **qmaster_params** is none.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

execd_params

This is used for passing additional parameters to the Altair Grid Engine execution daemon. The following values are recognized:

CL_WP_THREADS Defines the number of additional threads that are used in the sge_execd daemon for handling communication. This option will overwrite the value for "cl_threads" if defined in the bootstrap file line "communication_params" (See also *sge_bootstrap*(5) man page). Allowed values are a single number (e.g. cl_wp_threads=4) or a range specification

(e.g. cl_wp_threads=4-16). If this parameter is not set and there is also no wp_threads parameter set in the bootstrap file, the default value will be to use "cl_wp_threads=0". It is not supported to use more than 32 work pool threads.

ENABLE_DIR_SERVICE_TIMEOUT Enables a timeout that is used for operations that require a connection to a directory service (like NIS, LDAP, Active Directory, ...). Such operations are triggered by the execd and shepherd to retrieve user/group specific information to be able to start corresponding jobs. Valid timeout values are in the range between 1 and 10 seconds. Built-in default is 1 second and used if this parameter is not defined.

ENABLE_PVM_NOTIFY Enables sending signals from the sheperd process to the PVM tasker. Should only be enabled for those execution nodes where PVM jobs should be executed.

EXECD_RECONNECT_WAIT_TIME THIS PARAMETER IS NOT SUPPORTED AND MAINLY USED FOR TESTING PURPOSES! On qmaster shutdown an execd host will wait for some time before it will re-read the act_qmaster file and start to reconnect to the qmaster host. With this parameter it is possible to set the minimal waiting time before an execd tries to connect to the qmaster daemon again. The parameter defines the waiting time which can be set to a value from 1 to 30 seconds. The default value is 30 seconds. Changing this parameter will have immediate effect once the execd gets the new configuration.

HOST_PROVIDER This parameter can be used to tag the execution node as cloud based external node. It can be set to a freely selectable character string and currently has no further effect.

IGNORE_NGROUPS_MAX_LIMIT If a user is assigned to NGROUPS_MAX-1 supplementary groups so that Altair Grid Engine is not able to add an additional one for job tracking, the job will go into an error state when it is started. Administrators who want to prevent the system doing so can set this parameter. In this case the NGROUPS_MAX limit is ignored and the additional group (see gid_range) is not set. As a result for those jobs no online usage will be available. In addition, the parameter ENABLE_ADDGRP_KILL will have no effect. Please note that it is not recommended to use this parameter. Instead the group membership of the submit user should be reduced.

KEEP_ACTIVE If set to ERROR, the spool directory of the job (maintained by sge_shepherd(8)), the job script, a file which includes all job related messages from the execution daemon as also a list of all files located in the jobs temp-directory will be sent to the sge_qmaster(8) if the job had an exit-status != 0 or if the job failed (see sge_accounting(5)).

If set to ALWAYS, the execution daemon will send the spool directory as also the debugging files for every job.

These files can be found at '$SGE_ROOT/$SGE_CELL/faulty_jobs'

If set to true, the execution daemon will not remove the spool directory maintained by sge_shepherd(8) for a job (this value should only be set for debugging purposes).

If set to false, the execution daemon will remove the spool directory for a job. This also reduces writing of trace file messages to the shepherd trace file that is located in the spool directory during the time the job or its components are active.

KEEP_ACTIVE_SIZE If KEEP_ACTIVE is set to ERROR or ALWAYS the execution daemon will transfer files to sge_qmaster(8). As big files might lead to a high memory consumption in sge_qmaster(8), files with a bigger size than KEEP_ACTIVE_SIZE (in Bytes) will not get sent. If not set the file size is limited to 20 MB.

KEEP_OPEN_FDS As part of the start process for a job, the sge_shepherd will close all open file descriptors. If this should not be done for one or more file descriptors, this variable can be set to a number or to a range of numbers (like 4-9). Corresponding file descriptors will then not be closed. Please note that certain implementations of services (like Active Directory) require this functionality, so that the processes that are part of the job can use underlying library functionality.

LOG_RESOLVING_TIME This parameter can be used to enable time information logging in the qmaster messages file for hostname or IP address lookup operations. The parameter can be specified as a decimal number in seconds (e.g. "0.01"). When the parameter is not set or it is set to a negative value, no logging will be done. When the parameter is set to 0, logging will be enabled and the measured time for any resolving lookup will be logged as an INFO message. The INFO logging is a forced logging and will also be done if the current logging level is not set to INFO or higher. When the parameter is set to a value > 0, the measured time will be logged only if the given threshold has been reached. The resulting logging information will be printed as a WARNING message. When the parameter is not set or it is set to a negative value only lookups lasting more than 15 seconds will be logged as a WARNING message.

MAX_IJS_CLIENT_WAIT_TIME This parameter defines the waiting time for built-in interactive job support jobs to flush job output at job end and deliver the exit state to the connected qrsh or qlogin client. Once an interactive job has finished the sge_shepherd(8)) will wait for an acknowledge from the connected qrsh client. The already finished job will be shown as running during this time. If the qrsh client component is suspended for some reason or the network has some outage it might not always be useful to wait until the data is acknowledged.

If set to INFINITY the sge_shepherd(8)) will only finish on connection errors. If specified as a time value in the format HH:MM:SS all values >= 00:00:01 are used as timeout values. The default value for this parameter is 1 minute (00:01:00).

Once the timeout occurs the job will be reported as finished and the final exit state of the job is available in the accounting file.

The MAX_IJS_CLIENT_WAIT_TIME parameter has no influence on the suspend state of the qrsh client. A suspended qrsh client which was used to submit the job will stay suspended until the user unsuspends the qrsh.

NOTE: The job control feature of the shell where the qrsh job is submitted may be responsible for suspending the qrsh when it tries to read from stdin or tries to write to stdout if qrsh is started in the background. This can be bypassed using the qrsh -bgio parameter when submitting the qrsh job as a background job (see -bgio option of submit(1) man page for more information).

PTF_MIN_PRIORITY, PTF_MAX_PRIORITY The maximum/minimum priority which Altair Grid Engine will assign to a job. Typically this is a negative/positive value in the range of -20 (maximum) to 19 (minimum) for systems which allow setting of priorities with the nice(2) system call. Other systems may provide different ranges.
The default priority range (which varies from system to system) is installed either by removing the parameters or by setting a value of -999.
See the "messages" file of the execution daemon for the predefined default value on your hosts. The values are logged during the startup of the execution daemon.

PROF_EXECD This parameter is used to enable or disable profiling for an execution daemon (e.g. PROF_EXECD=true). Profiling provides the user with the ability to get system measurements. This can be useful for debugging or optimization of the system. The profiling output will be written to the messages file. The sge_execd profiling prints out information about e.g. host resolving times, time spent in communication or spooling, and many other areas.

SCRIPT_TIMEOUT This parameter defines the timeout value for scripts that are executed by sge_shepherd(8) (e.g. prolog/epilog of a job). Scripts where the execution duration would exceed the configured timeout value will be terminated by sge_shepherd(8) automatically. The default for this parameter is 2 minutes (00:02:00). It can be set to any value greater than 0.

NOTIFY_KILL This parameter allows you to change the notification signal for the signal SIGKILL (see -notify option of qsub(1)). The parameter accepts either signal names (use the -l option of kill(1)) or the special value none. If set to none, no notification signal will be sent. If it is set to TERM, for instance, or another signal name, this signal will be sent as a notification signal.

NOTIFY_SUSP With this parameter it is possible to modify the notification signal for the signal SIGSTOP (see -notify parameter of qsub(1)). The parameter accepts either signal names (use the -l option of kill(1)) or the special value none. If set to none, no notification signal will be sent. If it is set to TSTP, for instance, or another signal name, this signal will be sent as a notification signal.

USE_QIDLE This is a parameter used to start the experimental "qidle" load sensor. The parameter is no longer actively supported and may no longer be available in newer versions. If the parameter is set to true the qidle loadsensor binary is started. Default setting is "false". The "qidle" binary is not part of the standard distribution.

USE_QSUB_GID If this parameter is set to true, the primary group ID active when a job was submitted will be set to become the primary group ID for job execution. If the parameter is not set, the primary group ID as defined for the job owner in the execution host sgepasswd(5) file is used.
The feature is only available for jobs submitted via qsub(1), qrsh(1), and qmake(1). Also, it only works for qrsh(1) jobs and qmake(1)) if rsh and rshd components are used which are provided with Altair Grid Engine (i.e., the **rsh_daemon** and **rsh_command** parameters may not be changed from the default).

S_DESCRIPTORS, H_DESCRIPTORS, S_MAXPROC, H_MAXPROC, S_MEMORYLOCKED, H_MEMORYLOCKED, S_LOCKS, H_LOCKS Specifies soft and hard resource limits as implemented by the *setrlimit*(2) system call. See this manual page on your system for more information. These parameters complete the list of limits set by the RESOURCE LIMITS parameter of the queue configuration as described in *sge_queue_conf*(5). Unlike the resource limits in the queue configuration, these resource limits are set for every job on this execution host. If a value is not specified, the resource limit is inherited from the execution daemon process. Because this would lead to unpredicted results, if only one limit of a resource is set (soft or hard), the corresponding other limit is set to the same value.

S_DESCRIPTORS and H_DESCRIPTORS specify a value one greater than the maximum file descriptor number that can be opened by any process of a job.
S_MAXPROC and H_MAXPROC specify the maximum number of processes that can be created by the job user on this execution host
S_MEMORYLOCKED and H_MEMORYLOCKED specify the maximum number of bytes of

virtual memory that may be locked into RAM. The value type is memory_specifier as described in the sge_types(1) manual page.
S_LOCKS and H_LOCKS specify the maximum number of file locks any process of a job may establish.
All of these values can be specified using the multiplier letters k, K, m, M, g and G; see sge_types(1) for details. For all of these values, the keyword "INFINITY" (which means RLIM_INFINITY as described in the setrlimit(2) manual page) can be used to set the resource limit to "unlimited".

INHERIT_ENV This parameter indicates whether the shepherd should allow the environment inherited by the execution daemon from the shell that started it to be inherited by the job it is starting. When true, any environment variable that is set in the shell which starts the execution daemon at the time the execution daemon is started will be set in the environment of any jobs run by that execution daemon, unless the environment variable is explicitly overridden, such as PATH or LOGNAME. If set to false, each job starts with only the environment variables that are explicitly passed on by the execution daemon, such as PATH and LOGNAME. The default value is true.

SET_LIB_PATH This parameter tells the execution daemon whether to add the Altair Grid Engine shared library directory to the library path of executed jobs. If set to true, and INHERIT_ENV is also set to true, the Altair Grid Engine shared library directory will be prepended to the library path which is inherited from the shell which started the execution daemon. If INHERIT_ENV is set to false, the library path will contain only the Altair Grid Engine shared library directory. If set to false, and INHERIT_ENV is set to true, the library path exported to the job will be the one inherited from the shell which started the execution daemon. If INHERIT_ENV is also set to false, the library path will be empty. After the execution daemon has set the library path, it may be further altered by the shell in which the job is executed, or by the job script itself. The default value for SET_LIB_PATH is false.

ENABLE_ADDGRP_KILL If this parameter is set, Altair Grid Engine uses the supplementary group IDs (see gid_range) to identify all processes which are to be terminated when a job is deleted, or when sge_shepherd(8) cleans up after job termination.

SUSPEND_PE_TASKS With this parameter set to TRUE, tasks of tightly integrated jobs get suspended and unsuspended when the job gets suspended or unsuspended. Some MPI implementations are known to fail when tasks get suspended; in case you are running such jobs set SUSPEND_PE_TASKS to FALSE and handle suspension/unsuspension through a suspend_method and resume_method. See sge_queue_conf(5)).

PDC_INTERVAL This parameter defines the interval at which the PDC (Portable Data Collector) is executed by the execution daemon. The PDC is responsible for enforcing the resource limits s_cpu, h_cpu, s_vmem, h_vmem (see sge_queue_conf(5)), and job usage collection. The parameter can be set to a time_specifier (see sge_types(1)), to **PER_LOAD_REPORT**, or to **NEVER**. If this parameter is set to **PER_LOAD_REPORT** the PDC is triggered at the same interval as **load_report_time** (see above). If this parameter is set to **NEVER** the PDC run is never triggered. The default value for this parameter is 5 seconds.**
Note:** A PDC run is quite compute-intensive and may degrade the performance of running jobs. But if the PDC runs less often or never the online usage can be incomplete or totally missing (for example online usage of very short running jobs might be missing) and the resource limit enforcement is less accurate or would not happen if PDC is turned off completely.

PDC_CACHE_ENABLE_TIMEOUT The execution daemon will re-read the process info from the process table until the specified timeout is reached. The reason is a short race condition that can happen when the sge_shepherd process is starting a child process for a job.

The time between starting the child process and setting the additional group id is short, but when the PDC is exactly running at this moment the process id would not be detected as an Altair Grid Engine process. Using 2 seconds difference between first read time and last update time should be enough to trigger a re-read of such Process IDs. This is also the default value for this parameter.

Hint: If a triggered PDC run after job starts is enabled by also setting the PDC_TRIGGER_AFTER_JOB_START parameter, the PDC_TRIGGER_AFTER_JOB_START value should be set to a larger value than the value defined by PDC_CACHE_ENABLE_TIMEOUT parameter.

Possible setting: 0 - off int value - time in seconds (default 2 sec)

PDC_CACHE_UPDATE_TIMEOUT This parameter is used to define the period of time that the cached process data of a process that was not identified to belong to an Altair Grid Engine started job is valid. If the timeout is not reached the cached proc table information will be used. Once the information for the process is older than the defined timeout it will be re-read from the proc table. If the parameter is set to "0" the cache refreshing is turned off. This means once the process information is cached there will never be an update to the cached information.
If you have a high throughput of jobs in your Altair Grid Engine cluster and your process ID roll-over time is short it is recommended to set this parameter to a value below your typical process ID wrap-around time.
The default for this parameter is 120 seconds. This means that the cached process information for such processes is updated every two minutes.

PDC_TRIGGER_AFTER_JOB_START

This parameter is used for calculation of a triggered PDC run when jobs are started. It defines the time in seconds when a triggered PDC should be done when a new job was started.

Hint: If this parameter is set, it should be set to a value larger than PDC_CACHE_ENABLE_TIMEOUT. If this is not the case the re-read of Altair Grid Engine process IDs not yet detected as part of an Altair Grid Engine job from the process table and therefore the detection if it belongs to an Altair Grid Engine job is done in the standard PDC interval.

Possible setting: Keyword NEVER - off (default) int value - time in seconds

PDC_PROC_WARN_TIME Job usage retrieval in sge_execd on Linux is done by reading information from the /proc filesystem. Due to an issue in various Linux kernels opening files in /proc can take a significant time. When opening a file in /proc takes longer than the threshold defined by PDC_PROC_WARN_TIME in milliseconds a warning is generated in the messages file of the sge_execd daemon. Default is 10 milliseconds. By setting PDC_PROC_WARN_TIME to 0 monitoring of the time required for the open calls can be disabled.

LOG_PDC_TIME This parameter can be used to enable profiling logging in the execd messages file for /proc file parsing operations.

When the parameter is not set or it is set to a negative value, no profiling will be done.

When the parameter is set to 0, profiling will be done and the resulting time needed will always be logged as an INFO message.

When the parameter is set to a value (threshold specified in seconds as decimal number) > 0, profiling will be done; if the needed parsing time exceeds the given threshold a WARNING message will be logged.

The message logging will be forced and will also be done if current loglevel is not set to INFO or WARNING.

ENABLE_BINDING If this parameter is set, Altair Grid Engine enables the core binding module within the execution daemon to apply binding parameters that are specified during submission of a job. This parameter is not set by default and therefore all binding-related information will be ignored for hosts other than lx-amd64 and lx-x86. If the host has such an lx-amd64 or lx-x86 architecture it is internally turned on by default. Find more information for job-to-core binding in the section binding of qsub(1).

DISABLE_GID_RANGE_OBSERVATION If this parameter is set to 1 (or true), gid range observation is turned off in the Altair Grid Engine execution daemon. The default for this option is 0 (or false) which means the execd will by default the processes running on the execution host. If a process is using a group ID that is reserved for starting Altair Grid Engine jobs and it does not belong to a current running Altair Grid Engine job this group ID will be blocked for starting further Altair Grid Engine jobs until the unexpected processes are gone.

DISABLE_M_MEM_FREE If this parameter is set to 1 (or true), the execution daemon does not report load values for m_mem_free anymore. This is needed especially in cases where resource reservation for jobs requesting such a complex value must be enabled. When a load value of a specific host is lower than the requested value for a job, the scheduler does no resource reservation for that host. In order to prevent this the load value reporting can be turned off.

DISABLE_EXT_LS_WAIT If this parameter is set to true the execd will not wait for external loadsensors to report their first result before sending a full load report. This means the execution daemon will stay in unknown state until all external loadsensor scripts have reported the load values. Default value for this parameter is “false” and this will ensure that the scheduler will not schedule jobs based on wrong or missing load values. If the loadsensor script is running very long to obtain the reported information it might be useful to modify the load sensor script to report “safe” data on the first startup and provide the real measured data in the next run. This would also improve the execd startup time. If the loadsensor values provided by the external script can not lead to unwanted scheduling decisions this parameter might be turned off to improve execd startup time.

ENABLE_MEM_DETAILS If this parameter is set to 1 (or true) execution daemons on Linux report additional per job memory usage: pss (proportional set size), smem (shared memory), pmem (private memory), maxpss (maximum proportional set size).

The values for pss, smem and pmem are retrieved by the Linux sge_execd by summing up values read per process from the /proc//smaps file, see proc(5) e.g. from http://man7.org/linux/man-pages/man5/proc.5.html

maxpss is calculated in sge_execd by building the maximum from the pss values reported per load report interval.

These additional memory usage values can be retrieved via qstat -j .

ENFORCE_LIMITS When a job is started by sge_execd(8) limits configured in the sge_queue_conf(5) or specified during job submission will be set as per process resource limit, see also setrlimit(2).

The following limits are in addition enforced by sge_execd as per job limits: h_cpu, s_cpu, h_rss, s_rss, h_vmem, s_vmem. If **cgroups_params** is set to true h_vmem is controlled only by cgroups (see **cgroups_params** for more information).

The ENFORCE_LIMITS parameter allows the specification of where the limits h_cpu, s_cpu, h_rss, s_rss, h_vmem, s_vmem are enforced. This parameter has no influence on the limits h_stack, s_stack, h_data, s_data, h_core, s_core, h_fsize and s_fsize which are always set as resource limit with setrlimit() system call.

If ENFORCE_LIMITS is not set or if it is set to the value ALL the limits are both set as resource limit and they are enforced by sge_execd(8) as well.

If ENFORCE_LIMITS is set to SHELL, only the resource limits are set, sge_execd(8) will not enforce them.

If ENFORCE_LIMITS is set to EXECD, these limits are only enforced by sge_execd(8) INFINITY is set as resource limit.

If ENFORCE_LIMITS is set to OFF, sge_execd(8) will not enforce them and INFINITY is set as resource limit.

MONITOR_PDC When this parameter is set to true sge_execd(8) will write information and errors reported by the data collector to its messages file, e.g. errors when reading from the /proc file system. MONITOR_PDC can be set to 0 (or false) or 1 (true). Use this parameter with care and only when suggested by AGE Support, e.g. for debugging issues with the reporting of online usage, as a significant amount of information might get written to the sge_execd(8) messages file.

JOB_START_FLUSH_DELAY When a job is started by sge_execd(8) a job report will be sent to sge_qmaster(8) to trigger the job state transition from transferring to running. For short running jobs (runtime of a few seconds) the sending of the first job report can be delayed via the JOB_START_FLUSH_DELAY parameter. It specifies in seconds how long sending of the first job report will be delayed. Valid values for JOB_START_FLUSH_DELAY are 0 to 10, default is 0 (first job report will be sent immediately after receipt of job). Delaying the sending of the first job report can reduce load on sge_qmaster when many short jobs are run in the cluster.

RESCHEDULE_ON_KILLED_EPILOG If set to “true” or “1”, the behaviour depending on the exit status of the epilog as described in section epilog is in effect, which means if the epilog dies because of a signal, causing its exit_status to be larger than 127, the queue is put in error state and the job is re-queued to the pending job list.
If set to “false” or “0”, if the epilog dies because of a signal, the job finishes normally and the queue is not put in error state. The “failed” field of the job is set to “15 : in epilog” then, but the “exit_status” is the one of the job itself.
To detect if an epilog was signaled solely its exit status is taken into account, i.e. an epilog that exits with a status > 127 is handled like an epilog that was signaled.
The default value of this parameter is “true”.

RESCHEDULE_ON_MISSING_EPILOG If set to “true” or “1”, the behaviour depending on the exit status of the epilog as described in section *epilog* is in effect, which means if the epilog is configured but the epilog script cannot be found, the queue is put in error state and the job is re-queued to the pending job list.
If set to “false” or “0”, if the epilog is configured but the epilog script cannot be found, the job finishes normally and the queue is not put in error state. The “failed” field of the job is

set to “15 : in epilog” then, but the “exit_status” is the one of the job itself.
The default value of this parameter is “true”.

START_CONTAINER_AS_ROOT If set to “true” or “1”, for Docker jobs that start the sge_container_shepherd as the entrypoint of the Docker containers, the containers are started as user “root”. This allows the sge_container_shepherd to start pre and post scripts like prolog and pe_start as a different user than the job user.
If set to “false” or “0”, these containers are started as the job user.
For Docker jobs that do not specify a job script and therefore use the entrypoint that is defined in the Docker image of the container, the parameter START_CONTAINER_AS_ROOT is ignored.
The default value of this parameter is “false”.

AUTOMAP_CONTAINER_USERS This parameter is ignored if *START_CONTAINER_AS_ROOT* is set to “false” or “0”.
If *AUTOMAP_CONTAINER_USERS* is set to “TEMPORARY”, the Grid Engine admin user, the job user and all users of the pre and post commands “prolog”, “pe_start”, “per_pe_start_prolog”, “per_pe_start_epilog”, “pe_stop” and “epilog” are automatically mapped into the Docker container of the job. “mapped into” means for each of these users the user ID and the primary group ID are resolved on the execution host and are transferred to the container. For all jobs that start the “sge_container_shepherd” as the entrypoint of the container, this “sge_container_shepherd” then does not use the system calls to get the user ID and primary group ID of these users, instead it uses the transferred IDs. This way the “sge_container_shepherd” can start all these scripts and the job as the configured users even if these users are not defined inside the container.
But for the scripts themselves and for the job, these users still do not exist inside the container. In order to be able to run scripts and jobs that have to lookup the user ID and primary group ID of the user they run as, the *AUTOMAP_CONTAINER_USERS* parameter can also be set to “PERSISTENT”, which causes the “sge_container_shepherd” to write an entry to the “/etc/passwd” file for all these users. With this entry, a job that tries to read the user information from the system will get a valid answer, but this does not allow to switch to this user, i.e. running “su - ” will still fail. If AUTOMAP_CONTAINER_USERS=PERSISTENT is configured, Altair Grid Engine prevents the start of jobs that map files to “/etc/passwd” or “/etc/group” in the container.
The default value of this parameter is “false”.

CONTAINER_PE_HOSTFILE_COMPLEX If a parallel Docker job is started where the container hostnames are selected from RSMAPs, the execution daemon of the master task writes a “container_pe_hostfile” with all the container hostnames in the “pe_hostfile” format if the “execd_params” “CONTAINER_PE_HOSTFILE_COMPLEX” is set to the name of the RSMAP complex that defines the hostnames.
E.g.:
If there is a RSMAP “cont_hosts” declared and on each execution host it defines values like:

```
cont_host=4(host1_cont1 host1_cont2 host1_cont3 host1_cont4)
```

and a job is submitted using

```
qsub -pe mype 4 -l docker,docker_images="*image:latest*",cont_host=1 job_script.sh
```

and the scheduler decides to schedule the master task to host1, two slave tasks to host2 and one slave task to host3, the "container_pe_hostfile" might contain:

```
host1_cont3 1 <NULL> <NULL>
host2_cont1 1 <NULL> <NULL>
host2_cont4 1 <NULL> <NULL>
host3_cont2 1 <NULL> <NULL>
```

This allows to read this information in a per_pe_task_prolog and set the hostnames of the containers inside of the containers accordingly.
The default value for this parameter is an empty string.

DOCKER_RESPONSE_TIMEOUT This parameter defines how many seconds Altair Grid Engine waits for a response to a request it sent to the Docker daemon. If no response is received within this time, the Docker daemon is considered to be down and Grid Engine reacts accordingly, e.g. sets the job in error state etc. The response does not have to be complete in order to match the timeout, each single character received from the Docker daemon resets the timeout counter, but the time between two characters may never be longer than the "*DOCKER_RESPONSE_TIMEOUT*".
The minimum value is "10", the maximum is "86400" (one day), any value outside this range is ignored and the default value is used instead.
The default value of this parameter is "60".

DOCKER_SKIP_IMAGE_CHECK For Docker jobs sge_execd checks if the requested image is already existing on the execution host. If the image does not exist jobs having a hard docker_images request will not be started. For jobs having a soft docker_images request sge_execd will trigger an automatic download of the image.

If the parameter DOCKER_SKIP_IMAGE_CHECK is set to 1 or TRUE sge_execd will skip the check for hard requested docker_images requests. For jobs having a soft docker_images request it will still trigger the download of a non existing image.

This can be used in cases where we might not want to rely on the docker_images load value, e.g. as a very high load report interval is configured in a huge cluster. In this case the images existing on a host can be configured in the exec hosts complex_values, by setting DOCKER_SKIP_IMAGE_CHECK to 1 or TRUE we avoid jobs being rejected by sge_execd as it might not yet know about the images itself.

DOCKER_CREDENTIALS_FILE Path of a file which was created by "docker login" and contains credentials to authenticate against a remote Docker registry. Ignored if not set, empty or set to **NONE**.
The default value is **NONE**.

PROF_TIME Specifies the time interval when the profiling information should be printed for sge_execd daemons where profiling is enabled (see PROF_EXECD parameter). If the value is set to 00:00:00 (default) the profiling output will be printed every 60 seconds. Any other timeout value will print out the profiling information at the specified time.

RSMAP_CHAIN_MODE This parameter controls in which way RSMAP ids are chained. If set to **TOPOLOGY**, all RSMAP complexes with topology masks are chained. This means that Altair Grid Engine will try to get the same ids for all requested RSMAP complexes with topology masks and the job will not start if not enough ids are available and free. The default is **NONE**, which means that no RSMAP complexes are chained.

SET_CUDA_VISIBLE_DEVICES If set to "true", the environment variable `CUDA_VISIBLE_DEVICES` will be exported for jobs which request RSMAPs that have the parameter **cuda_id** or **uuid** configured (see sge_resource_map(5)). If an assigned RSMAP ID has both **cuda_id** and **uuid** configured, **uuid** will be used to set `CUDA_VISIBLE_DEVICES`. Please note that setting `CUDA_VISIBLE_DEVICES` may change the order of the GPUs as they are seen by the application and lead to unexpected behaviour when cgroups device isolation is used. Therefore it is recommended to configure **uuid** for each RSMAP ID when `SET_CUDA_VISIBLE_DEVICES` is set to "true". The default value is "false".

SET_ZE_AFFINITY_MASK If set to "true", the environment variable `ZE_AFFINITY_MASK` will be exported for jobs which request RSMAPs that have the parameter **xpu_id** configured (see sge_resource_map(5)).
Please note that setting `ZE_AFFINITY_MASK` may change the order of the GPUs as they are seen by the application and lead to unexpected behaviour when cgroups device isolation is used. The default value is "false".

SET_ROCR_VISIBLE_DEVICES If set to "true", the environment variable `ROCR_VISIBLE_DEVICES` will be exported for jobs which request RSMAPs that have the parameter **amd_id** configured (see sge_resource_map(5)).
Please note that setting `ROCR_VISIBLE_DEVICES` may change the order of the GPUs as they are seen by the application and lead to unexpected behaviour when cgroups device isolation is used. The default value is "false".

DCGM_PORT If this parameter is set to a port number > 0 the sge_execd will try connect to a NVIDIA DCGM daemon on the local host using the given port number. It will retrieve information about NVIDIA GPUs being present in the host. If the parameter is not set or set to 0 (default), the NVIDIA DCGM integration will not be used.

JOB_UMASK With this parameter, the umask of all jobs and their output- and error-files running on this host can be set. This parameter can either be set on the global host, and/or on individual hosts, which then overwrites a global setting. This parameter will be OR'ed with either the default, or user-chosen umask value. It acts as the least restrictive umask possible. Default is 0022. See also submit(1).

LINUX_MOUNT_NAMESPACE If this parameter is set to TRUE under Linux with kernel versions supporting the unshare system call, corresponding prolog and epilog scripts can be used to bind mount the /tmp file system for every individual job. For other architectures this execd_param is ignored. (further info in $SGE_ROOT/util/resources/lns/README)

CGROUPS_USER_CONTROL With this parameter it is possible to have the job user to get ownership (chown) over the per-job cgroups under all the subsystems or just under the memory subsystem. If set to *ALL*, the job user gets ownership over all job cgroups under all the subsystems. If set to *MEMORY*, the job user gets ownership only over the job cgroups under memory subsystem. If set to *NONE* (default), the ownership is not transferred at all to the job user.

NVML_LIBRARY_PATH This parameter sets a user specified path to the NVIDIA NVML library (libnvidia-ml.so) which will be used to query the number of GPU devices. This library will be preferred to the corresponding system library. The use of NVML can be disabled by setting *NVML_LIBRARY_PATH* to "none". Grid Engine will then use the PCI library to query the number of GPUs of each host (m_gpu).

SGE_GPU_QUERY_INTERVAL This parameter sets the minimum time interval in milliseconds between two consecutive queries to determine the number of GPUs available on the exe-

cution host. The maximum value is 43200000, which equals 12 hours. The default value is 300000, 5 minutes.

SUP_GRP_EVAL When this parameter is set to 0, execd writes SGE_SUP_GRP_EVAL=0 to the environment file in shepherd. This will be used in suppressing the evaluation of supplementary group ids, thereby not sending them to the qmaster for the Altair Grid Engine commands being run within a Altair Grid Engine job. No environment variable will be written when the param is set to 1. If set to -1, qmaster_params ENABLE_SUP_GRP_EVAL=FALSE takes precedence of writing SGE_SUP_GRP_EVAL=0 to the environment file. No environment variable will be written if ENABLE_SUP_GRP_EVAL=TRUE. Default value is set to -1.

DISABLE_SUP_GRP_EVAL_RANGE Contain gid range (also open-ended) which will be written to the shepherd environment file by execd, along with addgrpid as SGE_DISABLE_SUP_GRP_EVAL_RANGE. These values are ignored while sending supplementary group ids to the qmaster for the Altair Grid Engine commands being run within a Altair Grid Engine job.

ENABLE_BACKSLASH_ESCAPE This parameter defines how backslashes inside job arguments gets handled during the job start. If set to *no*, backslashes inside job arguments are never escaped. If set to *yes* (default), backslashes inside job arguments are properly escaped. If set to *auto*, the behavior will be same as when set to *no* for the following scenarios: When *shell_start_mode* is set to *posix_compliant* and queue's default shell (or the shell requested with -S) is (t)csh. When *shell_start_mode* is set to *unix_behavior* or *script_from_stdin* and if the job script is a (t)csh script. Setting environment variable, SGE_BACKSLASH_ESCAPE={no|yes|auto} will precedence and override the above param value. Note: (t)csh users must consider setting environment variable(preferably in users .cshrc), backslash_quote to avoid issues with escaping double-quotes in the arguments.

ENABLE_CGROUPS_USAGE_VALUES Enable or disable the collection of cgroups usage values. If enabled, cgroups memory usage values will be shown in qstat -j <job_id> and qacct -j <job_id>. The default value is **true**.

AGE_MISTRAL_MODE This parameter is used to set whether to enable/disable a job being profiled by Mistral.
If set to *DEFAULT_ON*, profiling via Mistral is enabled for all jobs to be run on the execution host. If set to *NEVER*, profiling via Mistral is disabled for all jobs to be run on the execution host.
If set to *IF_REQUESTED*, profiling via Mistral is enabled only for jobs requesting Mistral profiling with AGE_MISTRAL environment variable. (see ENVIRONMENTAL VARIABLES section in *qsub*(1)).
If set to *ALWAYS*, profiling via Mistral is enabled for all jobs irrespective of AGE_MISTRAL environment variable value). Default value for this param is set to *NEVER*.
Note: When both Breeze and Mistral are enabled, only Breeze profiling will happen.

AGE_MISTRAL_INSTALL_PATH This parameter is used to set the path where Mistral has been installed. This is a mandatory param for any job to be profiled by Mistral. This is also used to derive Mistral starter binary and library paths (if not already set in the config)

AGE_MISTRAL_LICENSE_PATH This parameter is used to set path to the license file for Mistral.

AGE_MISTRAL_LD_PRELOAD This parameter is used to set LD_PRELOAD library which will be used by Mistral for profiling. This is an optional param. If not set when profiling is requested, AGE will derive the library path relative to the Mistral installation path, `AGE_MISTRAL_INSTALL_PATH` as following:

For pre-2024.1.0 Mistral releases:
AGE_MISTRAL_INSTALL_PATH/dryrun/$LIB/libdryrun.so
For 2024.1.0 Mistral releases and later:
AGE_MISTRAL_INSTALL_PATH/$LIB/libmistral.so

AGE_MISTRAL_ENV This parameter is used to set path to the Mistral environment settings file. The file should contain name=value pairs for the environment variables which need to be set for a successfull execution of Mistral binaries. This is a mandatory param for any job to be profiled by Mistral.

AGE_BREEZE_MODE This parameter is used to set whether to enable/disable a job being profiled by Breeze.
If set to *DEFAULT_ON*, profiling via Breeze is enabled for all jobs to be run on the execution host.
If set to *NEVER*, profiling via Breeze is disabled for all jobs to be run on the execution host.
If set to *IF_REQUESTED*, profiling via Breeze is enabled only for jobs requesting Breeze profiling with AGE_BREEZE environment variable. (see ENVIRONMENTAL VARIABLES section in *qsub*(1)).
If set to *ALWAYS*, profiling via Mistral is enabled for all jobs irrespective of AGE_BREEZE environment variable value).
Default value for this param is set to *NEVER*.
Note: When both Breeze and Mistral are enabled, only Breeze profiling will happen.

AGE_BREEZE_INSTALL_PATH This parameter is used to set the path where Breeze has been installed. This is a mandatory param for any job to be profiled by Breeze. This is also used to derive the path to the Breeze starter method (if not already set in the config)

AGE_BREEZE_OUTPUT_DIRECTORY This parameter is used to set the directory path where Breeze write the output trace data. This is a mandatory param for any job to be profiled by Breeze.

AGE_BREEZE_STARTER_METHOD This parameter is used to set path to the Breeze launcher method. If not set when profiling is requested, AGE will derive the path relative to the Breeze installation path, `AGE_BREEZE_INSTALL_PATH` as following:
For pre-2024.1.0 Breeze releases:
AGE_BREEZE_INSTALL_PATH/trace-program.sh
For 2024.1.0 Breeze releases and later:
AGE_BREEZE_INSTALL_PATH/bin/trace-program.sh

IGNORE_EPILOG_CGROUPS_ERROR If set to "true", all cgroup related errors that occur after the job (before/after the epilog) will be ignored. The job will still have an exit status that relects the error, but the queue will not be put in error state and the job will not be rescheduled. The default value is **false**.

Changing **execd_params** will take effect after it was propagated to the execution daemons. The propagation is done in one load report interval. The default for **execd_params** is none.

The global configuration entry for this value may be overwritten by the execution host local configuration.

cgroups_params

A list of parameters for enabling and controlling the behavior of cgroups. This list can be set globally in the global configuration or be overridden in the host configuration for particular hosts. The cgroups feature is only available for lx-amd64 hosts. The OS must support cgroups (e.g. RHEL >= 6 with installed cgroup packages). Each cgroup subsystem must be mounted to a different subdirectory below **cgroup_path**. The following values are recognized:

cgroup_path If set to **none**, the cgroup support is disabled otherwise the path to the cgroup main directory is set here (usually /sys/fs/cgroup/cgroup). All cgroup subsystems must be available in subdirectories here (either as link or as mounted directories). Example: For memory limitation /sys/fs/cgroup/memory and for core binding /sys/fs/cgroup/cpuset must exist, when the cgroup_path is set to /sys/fs/cgroup.

subdir_name This is the name of the subdirectory within the cgroup subsystems containing job information. Default is SGE.service, which means that all cgroups will be created in /sys/fs/cgroup//SGE.service.

cpuset If set to **true** (or **1**), core binding is done by the cgroup cpuset subsystem. This affects only jobs requesting a core binding with the **-binding** submission parameter. Using cpuset is recommended since it limits the job to the chosen CPU cores without the possibility that the user overcomes these limits.

m_mem_free_limit_hard If set to **true** (or **1**), the usage of main memory is restricted to the value requested with **m_mem_free**. For using this parameter a mounted memory subsystem ($cgroup_path/memory) and a main memory request using the m_mem_free complex is required. If a job consumes more memory than requested it is usually aborted since further malloc calls will fail. Internally the value **memory.limit_in_bytes** is set. More details can be found in the operating system / cgroups documentation.

m_mem_free_limit_soft If set to **true** (or **1**) this parameter restricts the usage of main memory only if the memory limit is exceeded and the operating system detects memory contention. Main memory restriction is usually applied by pushing back the jobs main memory usage to the soft limits. Please consult the operating system / Linux kernel documentation for more details. Internally the value memory.soft_limit_in_bytes is set in the cgroup memory subsystem. If m_mem_free_limit_hard is active as well the hard memory limit rule is applied by the cgroup subsystem.

h_vmem_limit If set to **true** (or **1**) this parameter restricts the usage of the sum of main memory usage and swap space usage for a job if **h_vmem** is requested. Internally the cgroup parameter memory.memsw.limit_in_bytes is set. Please note that when the limit is applied successfully the **h_vmem** rlimit is not set for the job anymore. The execd daemon will also not enforce the h_vmem limit. This means only cgroups will handle the specified **h_vmem** rlimit. If the value is lower than **min_memory_limit** it is automatically increased to the configured amount. If **m_mem_free_limit_hard** is used and **m_mem_free** requested with a higher value than **h_vmem**, **m_mem_free** is reduced to **h_vmem** limit. If **m_mem_free** is set to a lower value than **h_vmem**, the kernel ensures that only **m_mem_free** main memory is available for the job, when requesting more memory it is automatically taken from swap space. Only when the total memory exceeds the **h_vmem** limit cgroups will do some action. If **h_vmem** is requested but no **m_mem_free**, automatically a hard cgroup limit for main memory with the size of **h_vmem** is applied, otherwise virtual memory limitation will not

work. This is a cgroup limitation. In this case a **min_memory_limit** value affects **h_vmem** as well.

min_memory_limit If set to a memory value (like 10M), each **m_mem_free** (or **h_vmem** request, when mixed with **m_mem_free**) which restricts the job due with **m_mem_free_limit_hard** or **m_mem_free_limit_soft** and which is lower than this value is automatically increased to the specified min_memory_limit value. Example: If **m_mem_free_limit_hard** is enabled and the job requests 100M but **min_memory_limit** is set to 150M, the internal limit for the job (memory.limit_in_bytes) is set to 150M. This does not affect qstat or internal book keeping. The parameter is used to solve OS specific issues with too large memory footprints (shepherd is part of the restriction) of small jobs. The memory is not multiplied by amount of slots requested by the job. The parameter is turned off by setting to 0 or not setting the parameter at all. Jobs just requesting cgroups **h_vmem** without **m_mem_free** are not affected. Here the same limits like for **h_vmem** are used.

freezer If set to **true** (or **1**) it enables the cgroup freezer subsystem for job suspension and resumption. The freezer subsystem needs to be mounted under $cgroup_path/freezer. If enabled a job is not longer suspended with the SIGSTOP and resumed with SIGCONT, the job is disabled from being scheduled by the Linux kernel by the freezer subsystem. There is no signal sent to the job. The processes are usually put in D state (which is an uninterruptible sleep, like for IO). If the job needs to be notified, the **-notify** submission option can be used. The queue configuration can override the cgroups suspension mechanism for certain jobs. This is done by putting the standard signals in the suspend_method (SIGSTOP) and resume_method (SIGCONT). This can be needed for certain job types which relay on signaling. For tightly integrated jobs only the master task is put into suspend state (the first task regardless JOB_IS_FIRST_TASK is configured in the parallel environment configuration or not). If all tasks of a parallel job has to be put in the freezer, **freeze_pe_tasks** needs to be activated. If queue overrides freezer with own signals, **freeze_pe_tasks** is set to true, but SUSPEND_PE_TASKS (**execd_params**) is set to false than slave tasks are not signaled. The freezer is available for batch and parallel jobs, but not for interactive jobs (**qlogin** and **qrsh**, except for **qrsh -inherit**).

freeze_pe_tasks If set to **true** (or **1**) and the freezer subsystem is turned on, not only the master task is suspended also all slave tasks of the parallel job are frozen. If queue overrides freezer with own signals and **freeze_pe_tasks** is set to true, but SUSPEND_PE_TASKS (**execd_params**) is set to false, slave tasks are not signaled. If SUSPEND_PE_TASKS is true (this is the default when not set as **execd_param**), slave tasks are signaled with the overridden queue signal / suspend_method.

killing If set to **true** (or **1**) the job is killed by using the tasks file of the cpuset subsystem (which when killing is enabled is automatically used for all jobs). As long as there are processes in the file the processes are signaled. This prevents any leftover processes from jobs to be running after the job finished.

mount Tries to mount the cgroup subsystems if it is not already mounted to $cgroup_path/subsystem before a cgroup is created. If $cgroup_path does not exist an error occurs (it will not tried to be created). If the subsystem directory does not exist it will be created. The subsystem is not unmounted by Grid Engine. Usually the mounting is done automatically by the operating system when it is started, so this parameter is usually turned off. Typically (like in RHEL 6) the configuration file for OS auto-mounting of cgroups is /etc/cgconfig.conf.

forced_numa When memory binding was requested with **-mbind cores:strict**, so that only

memory from the NUMA node the job is bound to (by using **-binding**) should be taken, this is set in the cgroups settings cpuset.mems. If turned on by setting **forced_numa** to **1** or **true**, this limit is ensured by the Linux kernel. In difference to the traditional memory enforcement the job cannot reset the value in order to get memory from other NUMA nodes.

devices A list of paths separated with "|". Access to these paths is blocked using cgroups and a job will not be able to read from or write to devices that are represented by these paths, unless access was granted with RSMAPs (see sge_resource_map.5). Ranges within brackets are be interpreted as numeric ranges, i.e. **devices=/dev/nvidia[0-254]|/dev/nvidiactl** will block access to all NVIDIA GPUs (/dev/nvidia0 to /dev/nvidia254) and to /dev/nvidiactl. If set to **none**, no devices will be blocked.

swappiness If set to **true** (or **1**), the "swappiness" of a job can be controlled with the builtin complex resource **cg_swappiness**. If requested by a job, the value of **cg_swappiness** is written to the cgroups file **memory.swappiness** and will override the default value.

cgroups version 2 does not have a **memory.swappiness** file to control the swap behaviour of a job/process. If cgroups version 2 is used and **cg_swappiness** is set to 0, Altair Grid Engine will disable swap for the job by writing 0 to **memory.swap.max**. Setting **cg_swappiness** to a value >= 1 will not have any effects on the swap behavior of the job.

cpuacct If set to **true** (or **1**), the cpuacct (CPU accounting) subsystem/controller will be enabled for all jobs. The system will then start collecting reports for usage of CPU resources for the jobs. The value is reported as "cpu_usage" in qstat -j and as "cgroups_cpu_usage" in qacct -j.

report_mem_usage If set to **true** (or **1**), a memory cgroup is created for all jobs, regardless whether they request memory values or not. This allows to collect cgroups memory usage values for jobs that would normally not be added to the memory cgroup. The default cgroup limits are applied to the job. The default value is **true**.

mem_hardwall If set to **true** (or **1**), **1** will be written to the cgroups file **cpuset.mem_hardwall** for all jobs requesting binding. This way kernel allocations of memory page and buffer data are restricted to the assigned memory nodes. The default value is **false**.

Note: This parameter has no effect if cgroups version 2 is used.

memory_spread_page If set to **true** (or **1**), **1** will be written to the cgroups file **cpuset.mem_hardwall** for all jobs requesting binding. This way file system buffers are spread evenly accross the assigned memory nodes. The default value is **false**.

Note: This parameter has no effect if cgroups version 2 is used.

reporting_params

Used to define the behavior of reporting modules in the Altair Grid Engine qmaster. Changes to the **reporting_params** takes immediate effect. The following values are recognized:

accounting If this parameter is set to true, the accounting file is written. The accounting file is prerequisite for using the **qacct** command.

reporting If this parameter is set to true, the reporting file is written. The reporting file contains data that can be used for monitoring and analysis, like job accounting, job log,

host load and consumables, queue status and consumables and sharetree configuration and usage. Attention: Depending on the size and load of the cluster, the reporting file can become quite large. Only activate the reporting file if you have a process running that will consume the reporting file! See sge_reporting(5) for further information about format and contents of the reporting file.

newjobrecord The "newjobrecord" parameter only has an effect if reporting is switched on. If the parameter is set to "true", data records for new jobs are written to the reporting file. Default setting for this parameter is "false". If the parameter is not specified, the value "false" is assumed.

flush_time Contents of the reporting file are buffered in the Altair Grid Engine qmaster and flushed at a fixed interval. This interval can be configured with the flush_time parameter. It is specified as a time value in the format HH:MM:SS. Sensible values range from a few seconds to one minute. Setting it too low may slow down the qmaster. Setting it too high will make the qmaster consume large amounts of memory for buffering data.

accounting_flush_time Contents of the accounting file are buffered in the Altair Grid Engine qmaster and flushed at a fixed interval. This interval can be configured with the accounting_flush_time parameter. It is specified as a time value in the format HH:MM:SS. Sensible values range from a few seconds to one minute. Setting it too low may slow down the qmaster. Setting it too high will make the qmaster consume large amounts of memory for buffering data. Setting it to 00:00:00 will disable accounting data buffering; as soon as data is generated, it will be written to the accounting file. If this parameter is not set, the accounting data flush interval will default to the value of the flush_time parameter.

max_submit_cmd_length Long command lines might result in quickly growing accounting and reporting files. This parameter allows to limit the maximum size of the submit command line that is stored as part of those file records. If this parameter is specified it defines the maximum amount of bytes that will be stored from the submit command line. Command lines that exceed that limit will get truncated. In case of absence of this parameter the limit will be set to 2000KB.

joblog If this parameter is set to true, the reporting file will contain job logging information. See sge_reporting(5) for more information about job logging.

sharelog The Altair Grid Engine qmaster can dump information about sharetree configuration and use to the reporting file. The parameter sharelog sets an interval in which sharetree information will be dumped. It is set in the format HH:MM:SS. A value of 00:00:00 configures qmaster not to dump sharetree information. Intervals of several minutes up to hours are sensible values for this parameter. See sge_reporting(5) for further information about sharelog.

online_usage Online usage information of running jobs (e.g. cpu, mem, vmem, ...) can be written to the sge_reporting(5) file. Which variables to report is configured as a colon separated list, e.g. online_usage=cpu:mem:vmem.

finished_jobs

Note: Deprecated; may be removed in a future release.
Altair Grid Engine stores a certain number of just finished jobs to provide post mortem status information. The **finished_jobs** parameter defines the number of finished jobs stored.

If this maximum number is reached, the eldest finished job will be discarded for every new job added to the finished job list.

Changing **finished_jobs** will take immediate effect. The default for **finished_jobs** is 0.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

qlogin_daemon

This parameter specifies the mechanism that is to be started on the server side of a qlogin(1) request. Usually this is the builtin mechanism. It's also possible to configure an external executable by specifying the full qualified pathname, e.g. of the system's telnet daemon.

Changing **qlogin_daemon** will take immediate effect. The default value for **qlogin_daemon** is builtin.

The global configuration entry for this value may be overwritten by the execution host local configuration.

Examples for the two allowed kinds of attributes are:

```
qlogin_daemon builtin
```

or
qlogin_daemon /usr/sbin/in.telnetd

qlogin_command

This is the command to be executed on the client side of a qlogin(1) request. Usually this is the builtin qlogin mechanism. It's also possible to configure an external mechanism, usually the absolute pathname of the system's telnet client program. It is automatically started with the target host and port number as parameters.

Changing **qlogin_command** will take immediate effect. The default value for **qlogin_command** is builtin.

The global configuration entry for this value may be overwritten by the execution host local configuration.

Examples for the two allowed kinds of attributes are:

```
qlogin_command builtin
```

or
qlogin_command /usr/bin/telnetd

rlogin_daemon

This parameter specifies the mechanism that is to be started on the server side of a qrsh(1) request **without** a command argument to be executed remotely. Usually this is the builtin mechanism. It's also possible to configure an external executable by specifying the absolute pathname, e.g. of the system's rlogin daemon.

Changing **rlogin_daemon** will take immediate effect. The default for **rlogin_daemon** is builtin.

The global configuration entry for this value may be overwritten by the execution host local configuration.

The allowed values are similar to the ones of the examples of **qlogin_daemon.**

rlogin_command

This is the mechanism to be executed on the client side of a qrsh(1) request **without** a command argument to be executed remotely. Usually this is the builtin mechanism. If no value is given, a specialized Altair Grid Engine component is used. The command is automatically started with the target host and port number as parameters. The Altair Grid Engine rlogin client has been extended to accept and use the port number argument. You can only use clients, such as ssh, which also understand this syntax.

Changing **rlogin_command** will take immediate effect. The default value for **rlogin_command** is builtin.

The global configuration entry for this value may be overwritten by the execution host local configuration.

rsh_daemon

This parameter specifies the mechanism that is to be started on the server side of a qrsh(1) request **with** a command argument to be executed remotely. Usually this is the builtin mechanism. If no value is given, a specialized Altair Grid Engine component is used.

Changing **rsh_daemon** will take immediate effect. The default value for **rsh_daemon** is builtin.

The global configuration entry for this value may be overwritten by the execution host local configuration.

rsh_command

This is the mechanism to be executed on the client side of a qrsh(1) request **with** a command argument to be executed remotely. Usually this is the builtin mechanism. If no value is given, a specialized Altair Grid Engine component is used. The command is automatically started with the target host and port number as parameters like required for telnet(1) plus the command with its arguments to be executed remotely. The Altair Grid Engine rsh client

has been extended to accept and use the port number argument. You can only use clients, such as ssh, which also understand this syntax.

Changing **rsh_command** will take immediate effect. The default value for **rsh_command** is builtin.

The global configuration entry for this value may be overwritten by the execution host local configuration.

port_range

This parameter is used to define fix port ranges that are used by the interactive job execution modules ("builtin" or "daemon" based). The builtin and the "daemon" based components will bind TCP/IP ports within the specified port range. The use case would be to setup an open port range in a firewall configuration.

If there is no free port available in the specified range the interactive command will fail. The configured range should be large enough to handle all interactive clients (qrsh, qlogin, qsh, ...) that might be started on a host at the same time.

The "daemon" based interactive job methods will also bind ports on the execution host. The "builtin" method will only bind ports on the client side where the interactive command is started.

Changing this parameter will have immediate effect. For the "daemon" based interactive job methods the execd must get the new config first which happens usually within the next load report interval.

If no value is given (port_range=none) the resulting port is provided by the operating system.

The global configuration entry for this value may be overwritten by a local configuration.

The default value for **port_range** is "none".

The syntax for this parameter is:

```
none|port_range[,port_range,...]
```

Where port_range is defined as:

```
port_nr[-port_nr]
```

Example:

```
port_range 30400-30800,40400-40800
```

delegated_file_staging

This flag must be set to "true" when the prolog and epilog are ready for delegated file staging, so that the DRMAA attribute 'drmaa_transfer_files' is supported. To establish delegated file staging, use the variables beginning with "$fs_..." in prolog and epilog to move the input, output and error files from one host to the other. When this flag is set to "false", no file staging is available for the DRMAA interface. File staging is currently implemented only via the DRMAA interface. When an error occurs while moving the input, output and error files, return error code 100 so that the error handling mechanism can handle the error correctly. (See also FORBID_APPERROR).

reprioritize

Note: Deprecated; may be removed in a future release.
This flag enables or disables the reprioritization of jobs based on their ticket amount. The **reprioritize_interval** in sge_sched_conf(5) takes effect only if **reprioritize** is set to true. To turn off job reprioritization, the **reprioritize** flag must be set to false and the **reprioritize_interval** to 0 which is the default.

This value is a global configuration parameter only. It cannot be overridden by the execution host local configuration.

jsv_url

This setting defines a server JSV instance which will be started and triggered by the sge_qmaster(8) process. This JSV instance will be used to verify job specifications of jobs before they are accepted and stored in the internal master database. The global configuration entry for this value cannot be overwritten by execution host local configurations.

Find more details concerning JSV in sge_jsv(5) and sge_request(5).

The syntax of the **jsv_url** is specified in sge_types(1).

jsv_allowed_mod

If there is a server JSV script defined with **jsv_url** parameter, all qalter(1) or qmon(1) modification requests for jobs are rejected by qmaster. With the **jsv_allowed_mod** parameter an administrator has the possibility to allow a set of switches which can then be used with clients to modify certain job attributes. The value for this parameter has to be a comma-separated list of JSV job parameter names as they are documented in qsub(1) or the value **none** to indicate that no modification should be allowed. Please note that even if **none** is specified the switches **-w** and **-t** are allowed for qalter.

libjvm_path

libjvm_path is usually set during qmaster installation and points to the absolute path of libjvm.so. (or the corresponding library depending on your architecture -

e.g. /usr/java/jre/lib/i386/server/libjvm.so) The referenced libjvm version must be at least 1.5. It is needed by the JVM qmaster thread only. If the Java VM needs additional starting parameters they can be set in **additional_jvm_args**. If the JVM thread is started at all can be defined in the *sge_bootstrap*(5) file. If libjvm_path is empty or an incorrect path the JVM thread fails to start.

The global configuration entry for this value may be overwritten by the execution host local configuration.

additional_jvm_args

additional_jvm_args is usually set during qmaster installation. Details about possible values **additional_jvm_args** can be found in the help output of the accompanying Java command. This setting is normally not needed.

The global configuration entry for this value may be overwritten by the execution host local configuration.

gpu_job_usage

gpu_job_usage allows to configure which gpu specific job usage values will be reported by sge_execd and displayed in qstat -j <job_id>.

For more details see sge_amd(5), sge_nvidia(5) and sge_intel(5) (GPU Job Usage).

gpu_job_accounting

gpu_job_accounting allows to configure which gpu specific usage values will be written to the accounting file and displayed in qacct -j <job_id>.

For more details see sge_amd(5), sge_nvidia(5) and sge_intel(5) (GPU Job Accounting).

SEE ALSO

sge_intro(1), csh(1), qconf(1), qsub(1), sge__jsv(5), rsh(1), sh(1), getpwnam(3), drmaa_attributes(3), sge_queue_conf(5), sge_sched_conf(5), sge_types(1), sge_execd(8), sge_qmaster(8), sge_shepherd(8), sge_diagnostics(5), cron(8), Altair Grid Engine Installation and Administration Guide.

COPYRIGHT

sge_diagnostics

NAME

Diagnostics - Diagnostics and Debugging of Altair Grid Engine

DESCRIPTION

The sections below describe aspects of diagnosing qmaster behaviour and obtaining more detailed information about the state of Altair Grid Engine.

LOGGING

Certain components as sge_qmaster(8) or sge_execd(8) create informative, warning, error or debugging messages that are written to a message file of the corresponding component.

The parameter *loglevel* of the global configuration of Altair Grid Engine allows to change the level of information that is written to the message file. When the *loglevel* is set to *log_debug* then more detailed information is written that allows to see details of the internal state of the component and to debug certain error scenarios that would be difficult to diagnose otherwise.

Received and Sent Messages

When the *loglevel log_debug* is activated then Altair Grid Engine writes log messages whenever sge_qmaster receives messages or sends messages.

Message have the following format: ACTION: HOSTNAME/COMPROC-NAME/COMPROC-ID/MESSAGE-ID:MESSAGE-TAG:SIZE

- *ACTION*: SEND or RECEIVE
- *HOSTNAME*: Identifies the host were the message was send from.
- *COMPROC-NAME*: Name of the daemon or command that sent the message (e.g. qsub, execd, qmon, ...)
- *COMPROC-ID*: Altair Grid Engine internal ID used for communication
- *MESSAGE-ID*: Message ID that identifies the request on the communication layer.
- *MESSAGE-TAG*: Type of message: TAG_GDI_REQUEST, TAG_ACK_REQUEST, TAG_REPORT_REQUEST, ...
- *SIZE*: Size of the message in bytes

Request execution

When the *loglevel log_debug* is activated then Altair Grid Engine writes log messages whenever sge_qmaster accepts new requests from client commands (e.g qsub(1), qalter(1), qconf(1), ...), other server components (e.g. sge_execd) or qmaster internal threads (lothread when the Altair Grid Engine cluster is connected to Altair License Orchestrator). Incoming requests are stored in qmaster internal queues till a thread is available that is able to handle the request properly. Log messages will also be written when one of the internal qmaster threads start executing such a request and when request handling has finished.

In low performing clusters this allows to identify hosts, users, requests types ... that are the root cause for the performance decrease.

Messages related to request execution have following format:

```
ACTION: HOSTNAME/COMPROC-NAME/COMPROC-ID/MESSAGE-ID:USER:SIZE:INTERFACE:REQUEST-DETAILS[:DURAT
```

- *ACTION*: QUEUE, FETCHED, STARTED or FINISHED
- *HOSTNAME*: Identifies the host were the request was send from.
- *COMPROC-NAME*: Name of the daemon or command that sent the request (e.g. qsub, execd, qmon, ...)
- *COMPROC-ID*: Altair Grid Engine internal ID used for communication
- *MESSAGE-ID*: Message ID that identifies the request on the communication layer.
- *USER*: Name of the user that caused the request to be send to qmaster.
- *SIZE*: Size of the request in bytes (the commlib message) when receiving requests from external clients, else 0
- *INTERFACE*: Interface that was used to trigger the request (GDI or REP)
- *REQUEST-DETAILS*: For GDI requests this will show the operation type (e.g ADD, MOD, DEL, ...) and the object type (JB for job object, CQ for cluster queue object, ...)
- *DURATION*: optionally: Time in seconds since the last action on the request, e.g. time a request was queued, time it took from fetching a request till it can be processed (acquiring locks), time for processing a request
- *TASKS*: optionally: Amount of tasks part of GDI requests. 1 for other requests or for individual GDI requests
- *SIZE*: optionally: Size of the message in bytes

Messages related to non GDI requests modifying event clients (e.g. acknowledge receipt of an event package) have the following format:

```
ACTION(E): REQUEST:ID[:DURATION]
```

- *ACTION*: QUEUE, STARTED or FINISHED
- *REQUEST*: type of request, e.g. ACK
- *ID*: the event client id, see qconf -secl
- *DURATION*: optionally: time in seconds since the last action on the request, e.g. time a request was queued, time for processing a request

DATA COMPRESSION

Since Altair Grid Engine 8.6.0 it is possible to enable data compression at Altair Grid Engine communication layer. This chapter describes how compression can be enabled and verified.

ENABLE DATA COMPRESSION

Compression can be enabled by modifying or adding the line "communication_params" in the `$SGE_ROOT/$SGE_CELL/common/bootstrap` file. A detailed information about the syntax can be found at *sge_bootstrap*(5) man page.

Example: Enable zlib compression

The bootstrap file should get the following line:

```
communication_params msg_mode=zlib(min_len:5K,level:1)+none
```

Note: Once the bootsrap file has changed a restart of the Altair Grid Engine daemons must be done to make the changes effective.

This would enable zlib compression for messages larger than 5K with zlib compression level 1 (=best performance). All clients that do not support zlib compression still can connect to Altair Grid Engine daemons and using not compressed data transfer.

ENABLE DATA COMPRESSION FOR SINGLE CLIENTS

If data compression should only be used for special clients the bootstrap entry would look like this:

```
communication_params msg_mode=none+zlib(min_len:5K,level:1)
```

This will not enable zlib compression per default (the keyword "none" is important here). Clients that have set the environment variable **SGE_COMMLIB_COMMUNICATION_PARAMS** to overwrite the bootsrap setting might get compressed data transfer if the min_len setting matches for a data message.

If e.g. `qstat -j "*"` is started with env `SGE_COMMLIB_COMMUNICATION_PARAMS="msg_mode=zlib(min_len:5K,level` it will try to setup zlib compression when connecting to sge_qmaster daemon. An example can be found in the "VERIFY COMPRESSION SETUP" section below.

COMMLIB WORK POOL THREADS

The allowed number of additional threads used for communication inside sge_qmaster daemon can be defined in the "qmaster_params" parameter. The CL_WP_THREADS option will specify the behavior. The configured work pool threads are used by commlib in order to parallel handle incoming or outgoing messages. This parameter has an influence on communication performance (See *sge_conf*(5) man page).

VERIFY COMPRESSION SETUP

In order to verify if zlib compression is setup correctly the qping command can be used. All parameters of qping are described in the *qping*(1) man page. Here a short example showing client enabled data compression:

The qmaster was started with following bootstrap setting:

```
# cat $SGE_ROOT/default/common/bootstrap | grep communication
communication_params    msg_mode=none+zlib(min_len:0,level:1)
```

This means that the sge_qmaster daemon will not do zlib compression per default.

The following output shows a qstat client with and without environment variable

```
SGE_COMMLIB_COMMUNICATION_PARAMS="msg_mode=zlib(min_len:0,level:1)+none"
```

set and exported to the qping client binary:

```
sh-4.1# SGE_QPING_OUTPUT_FORMAT="h:2 h:5 h:6 h:7 h:8 h:9 h:11 h:14 w:4:10:18"
sh-4.1# export SGE_QPING_OUTPUT_FORMAT
sh-4.1# qping -dump -format bin -to qstat -from qstat walnut-vb $SGE_QMASTER_PORT qmaster 1
open connection to "walnut-vb/qmaster/1" ... no error happened
           time|d.|remote            |msg len|con count|msg ulen|msg cratio|msg mod|
---------------|--|------------------|-------|---------|--------|----------|-------|
12:27:58.619787|<-|walnut-vb/qstat/14|   3492|        6|    3492|     1.000|   none|
12:27:58.627035|->|walnut-vb/qstat/14| 458782|        6|  458782|     1.000|   none|
12:29:15.141192|<-|walnut-vb/qstat/15|    641|        6|    3492|     5.448|   zlib|
12:29:15.158727|->|walnut-vb/qstat/15|  13018|        6|  458782|    35.242|   zlib|
```

The first qstat did not compress the data. This can be verified by comparing in the “msg len”(=transferred data size) and the “msg ulen”(=uncompressed message length) values. The “msg cratio” column in addition shows the compression ratio.

It is important to say that compression will cost cpu performance, but strongly reduces network traffic. The resulting cluster performance depends on the message sizes, number of available cpus at qmaster, network performance and **CL_WP_THREADS** settings for the sge_qmaster daemon (See *sge_conf*(5) man page). The compression algorithm params “min_len” and “level” also have an influence on overall cluster performance (See *sge_bootstrap*(5) man page).

JOB LIMITS

SUPPORTED LIMITS

The following table shows what kind of limits are supported via job submission and queue setting and where the observation is implemented (sge_execd, sge_shepherd or via cgroups):

limit	execd	shepherd	cgroups	description
h_cpu/s_cpu	yes	yes	no	cpu time limit in seconds
h_vmem/s_vmem	yes*	yes*	yes	virtual memory size
h_rss/s_rss	yes*	yes*	no	resident set size
h_stack/s_stack	no	yes	no	stack size limit
h_data/s_data	no	yes	no	data segment size limit
h_core/s_core	no	yes	no	max. size of a core file**
h_fsize/s_fsize	no	yes	no	max. file size**

(* = If supported by OS) (** = This kind of limit is not adjusted on pe settings)

In order to setup limit observation by the sge_execd or sge_shepherd the "execd_params" parameter "ENFORCE_LIMITS" in the configuration of the execution hosts is used (see *sge_conf*(5) man page). This parameter only allows settings for the supported limits (cpu, vmem and rss). The remaining limits (stack, data, core and fsize) cannot be switched off by this parameter.

If virtual memory size is set to be observed by cgroups the sge_execd observation is disabled for the "h_vmem" limit. If cgroups limit setting did not report any error at the sge_shepherd startup the "h_vmem" resource limit will be set to "infinity" with the setrlimit() system call. How to enable cgroups "h_vmem" limit observation is described in the man page sge_conf(5) ("h_vmem_limit" parameter of "cgroups_params").

If a limit is observed by sge_execd the execd is responsible for killing the job. For sge_shepherd the limits are set via setrlimit() command to let the Kernel enforce the process limit. The cgroups implementation will write the corresponding limit for all processes of the job into the "memory.memsw.limit_in_bytes" file which is created in the cgroups directory of the job.

SUPPORTED LIMITS VIA EXECD CONFIGURATION

The following table shows the limits that can be set by sge_shepherd at job start via setrlimit() system call to enable Kernel enforced process limit. The OS must of course support this limit type.

Limit	Description
h_descriptors/s_descriptors	nr of open file descriptors
h_maxproc/s_maxproc	max nr of processes
h_locks/s_locks	nr of locks
h_memorylocked/s_memorylocked	maximum number of bytes of memory locked into RAM

Please see also the sge_conf(5) man page. The section "execd_params" contains information how to enable this limits.

GRANTED RESOURCES AND LIMIT ADJUSTMENT DEPENDING ON PE SETUP

For parallel jobs the granted resource value (which can be a limit value) that is applied to each parallel task depends on the used parallel environment (PE) settings.

There are parallel task specific resource requests and there are "job global" requests which apply to all parallel tasks without specific resource requests - with these exceptions: * If "job_is_first_task=false" is configured, the "job global" requests do not apply to the controller task (parallel task ID 0) of the parallel job.

Independently of the "job_is_first_task=false" exception, these two exception do apply: * If "master_forks_slaves=true" is configured, the resource requests of all tasks on the master host are added up and are applied to the controller task of the parallel job. No agent task on that host gets anything applied. * If "daemon_forks_slaves=true" is configured, the resource requests of all tasks on each non-master host are added up and are applied to the first (and only) task on that host. No further agent task on that gets anything applied.

Examples for master_forks_slaves=true in pe setting

PE mpi_jift_true defines allocation rule "2" and job_is_first_task=true.
PE mpi_jift_false defines allocation rule "2" and job_is_first_task=false.

- qsub -pe mpi_jift_true 4 -l h_vmem=1G
  PE task 0: host A, h_vmem = 2G
  PE task 1: host A
  PE task 2: host B, h_vmem = 1G
  PE task 3: host B, h_vmem = 1G
- qsub -pe mpi_jift_false 4 -l h_vmem=1G
  PE task 0: host A, h_vmem = 2G
  PE task 1: host A
  PE task 2: host A
  PE task 3: host B, h_vmem = 1G
  PE task 3: host B, h_vmem = 1G
- qsub -pe mpi_jift_false 4 -l h_vmem=1G -petask 0 -l h_vmem=3G
  PE task 0: host A, h_vmem = 5G
  PE task 1: host A
  PE task 2: host A
  PE task 3: host B, h_vmem = 1G
  PE task 3: host B, h_vmem = 1G

Examples for daemon_forks_slaves=true in pe setting

PE mpi_jift_true defines allocation rule "2" and job_is_first_task=true.
PE mpi_jift_false defines allocation rule "2" and job_is_first_task=false.

- qsub -pe mpi_jift_true 4 -l h_vmem=1G
  PE task 0: host A, h_vmem = 1G
  PE task 1: host A, h_vmem = 1G
  PE task 2: host B, h_vmem = 2G
  PE task 3: host B

- qsub -pe mpi_jift_false 4 -l h_vmem=1G
  PE task 0: host A
  PE task 1: host A, h_vmem = 1G
  PE task 2: host A, h_vmem = 1G
  PE task 3: host B, h_vmem = 2G
  PE task 4: host B
- qsub -pe mpi_jift_false 4 -l h_vmem=1G -petask 0 -l h_vmem=3G
  PE task 0: host A, h_vmem = 3G
  PE task 1: host A, h_vmem = 1G
  PE task 2: host A, h_vmem = 1G
  PE task 3: host B, h_vmem = 2G
  PE task 4: host B

Example: LIMIT ADJUSTMENT DEPENDING ON PE SETUP

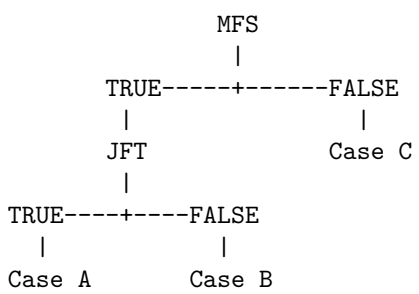
For parallel jobs without specific per task requests the resulting limit value depends on the used parallel environment (PE) settings. The following diagrams should explain this in more detail.

List of abbreviations:

Name	Description
agent limit	Value specified with “-l”
controller limit	Value specified with “-masterl” (or with -petask 0)
n	Nr of agent slots on this host
CS	Boolean “control_slaves” PE option
JFT	Boolean “job_is_first_task” PE option
MFS	Boolean “master_forks_slaves” PE option
DFS	Boolean “daemon_forks_slaves” PE option
n/a	Not applicable situation

Controller task limit adjustments

This diagrams shows how the resulting limit for the controller task of a parallel job is calculated:



Case A:

```
=======
              controller limit
                 requested?
                     |
       TRUE----------+---------FALSE
        |                        |
 controller limit +         agent limit +
(n-1) * agent limit       (n-1) * agent limit

Case B:
=======
              controller limit
                 requested?
                     |
       TRUE----------+---------FALSE
        |                        |
 controller limit +          agent limit +
  n * agent limit           n * agent limit

Case C:
=======
              controller limit
                 requested?
                     |
       TRUE----------+---------FALSE
        |                        |
 controller limit           agent limit
```

Adjustments for agent tasks running on master host

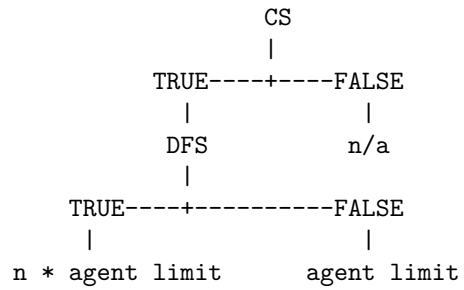
This diagrams shows the resulting limit for any agent task of a parallel job which is started on the controller task host:

```
                  CS
                   |
            TRUE----+----FALSE
             |             |
            MFS           n/a
             |
     TRUE---+---FALSE
      |           |
     n/a         DFS
                  |
            TRUE---+---FALSE
             |           |
            JFT      agent limit
             |
     TRUE---+----FALSE
      |            |
```

```
   (n-1) *         n *
 agent limit    agent limit
```

Adjustments for agent tasks running on a agent host

This diagrams shows the resulting limit for any agent task of a parallel job which is started on a agent host:



Examples for master_forks_slave=true in pe setting

```
qsub -l h_vmem=1G -pe mpi 3
h_vmem = 1G + 1G * 3 = 4G   (job first task=false)
h_vmem = 1G + 1G * 2 = 3G   (job first task=true)


qsub -masterl h_vmem=0.5G -l h_vmem=1G -pe mpi 3
h_vmem = 0.5G + 3 * 1G = 3.5G (job first task = false)
h_vmem = 0.5G + 2 * 1G = 2.5G (job first task = true)

qsub -pe fixed 16 -masterl h_vmem=64G
h_vmem = 64G + INFINITY * 16 = INFINITY (job first task = false)
h_vmem = 64G + INFINITY * 15 = INFINITY (job first task = true)

qsub -pe fixed 16 -masterl h_vmem=2G -l h_vmem=4G
h_vmem = 2G + 4G * 16 = 2G + 64G = 66G (job first task = false)
h_vmem = 2G + 4G * 15 = 2G + 60G = 62G (job first task = true)

qsub -pe fixed 16 -l h_vmem=4G
h_vmem = 4G + 4G * 16 = 4G + 64G = 68G (job first task = false)
h_vmem = 4G + 4G * 15 = 4G + 60G = 64G (job first task = true)
```

h_vmem limit for cgroups

Depending on the PE-setting, each PE agent task will get its individual limit, or the cgroups h_vmem limit will be the sum of the limits of all tasks started on this host (depending on the settings, this might include or exclude the master-task). If any adding occurs, once the individual limit for controller task, agent task on controller host and agent task on agent host are calculated the resulting sum for the cgroups h_vmem setting is done the following way:

```
On controller task host (controller task limit might be separate or added):
(resulting controller task limit +) nr of started agent tasks * resulting agent limit

On agent task host:
nr of started agent tasks * resulting agent limit
```

Note: The PE parameters daemon_forks_slaves and master_forks_slaves have also an influence on the nr of agent jobs that can be started on each host. More information about this parameters can be found in the sge_pe(5) man page.

MONITORING

MESSAGE FILE MONITORING

Monitoring output of the sge_qmaster(8) component is disabled by default. It can be enabled by defining *MONITOR_TIME* as *qmaster_param* in the global configuration of Altair Grid Engine (see sge_conf(5)). *MONITOR_TIME* defines the time interval when monitoring information is printed. The generated output provides information per thread and it is written to the message file or displayed with *qping(1)*.

The messages that are shown start with the name of a qmaster thread followed by a three digit number and a colon character (:). The number allows to distinguish monitoring output of different threads that are part of the same thread pool.

All counters are reset when the monitoring output was printed. This means that all numbers show activity characteristics of about one *MONITOR_TIME* interval. Please note that the *MONITOR_TIME* is only a guideline and not a fixed interval. The interval that is actually used is shown by **time** in the monitoring output.

For each thread type the output contains following parameters:

- **runs**: [iterations per second] number of cycles per second a thread executed its main loop. Threads typically handle one work package (message, request) per iteration.
- **out**: [messages per second] number of outgoing TCP/IP communication messages per second. Only those threads trigger outgoing messages that handle requests that were triggered by external commands or interfaces (client commands, DRMAA, ...).
- **APT**: [cpu time per message] average processing time per message or request.
- **idle**: [%] percentage how long the thread was idle and waiting for work.
- **wait**: [%] percentage how long the thread was waiting for required resources that where already in use by other threads (waiting for locks).
- **mutexwait**: [%] percentage how long the thread was waiting for required resources that where already in use by other threads (waiting for mutexes).
- **openmp** [%] time spent in OpenMP threads parallel to the actual thread execution.
- **gr** [%] percentage how long the thread was holding the global read lock.
- **gw** [%] percentage how long the thread was holding the global write lock.
- **time**: [seconds] time since last monitoring output for this thread was written.

Depending on the thread type the output will contain more details:

LISTENER

Listener threads listen for incoming messages that are send to qmaster via generic data interface, event client interface, mirror interface or reporting interface. Requests are unpacked and verified. For simple requests a response will also be sent back to the client but in most cases the request will be stored in one of the request queues that are processed by reader, worker threads or the event master thread.

- IN **g**: [requests per second] number of requests received via GDI interface.
- IN **a**: [messages per second] handled ack's for a request response.
- IN **e**: [requests per second] event client requests received from applications using the event client or mirror interface.
- IN **r**: [requests per second] number of reporting requests received from execution hosts.
- OTHER **wql**: [requests] number of pending read-write requests that can immediately be handled by a worker thread.
- OTHER **rql**: [requests] number of pending read-only requests that can immediately be handled by a reader thread.
- OTHER **wrql**: number of waiting read-only requests. read-only requests in *waiting*-state have to be executed as part of a GDI session and the data store of the read-only thread pool is not in a state to execute those requests immediately.

READER/WORKER

Reader and worker threads handle GDI and reporting requests. Reader threads will handle read-only requests only whereas all requests that require read-write access will be processed by worker threads.

- EXECD **l**: [reports per second] handled load reports per second.
- EXECD **j**: [reports per second] handled job reports per second.
- EXECD **c**: [reports per second] handled configuration version requests.
- EXECD **p**: [reports per second] handled processor reports.
- EXECD **a**: [messages per second] handled ack's for a request response.
- GDI **a**: [requests per second] handled GDI add requests per second.
- GDI **g**: [requests per second] handled GDI get requests per second.
- GDI **m**: [requests per second] handled GDI modify requests per second.
- GDI **d**: [requests per second] handled GDI delete requests per second.
- GDI **c**: [requests per second] handled GDI copy requests per second.
- GDI **t**: [requests per second] handled GDI trigger requests per second.

- GDI **p**: [requests per second] handled GDI permission requests per second.

EVENT MASTER

The event master thread is responsible for handling activities for registered event clients that either use the event client or the mirror interface. The interfaces can be used to register and subscribe all or a subset of event types. Clients will automatically receive updates for subscribed information as soon as it is added, modified or deleted within qmaster. Clients using those interfaces don't have the need to poll required information.

- **clients**: [clients] connected event clients.
- **mod**: [modifications per second] event client modifications per second.
- **ack**: [messages per second] handled ack's per second.
- **blocked**: [clients] number of event clients blocked during send.
- **busy**: [clients] number of event clients busy during send.
- **events**: [events per second] newly added events per second.
- **added**: [events per second] number of all events per second.
- **skipped**: [events per second] ignored events per second (because no client has subscribed them).

TIMED EVENT

The timed event thread is used within qmaster to either trigger activities once at a certain point in time or in regular time intervals.

- **pending**: [events] number of events waiting that start time is reached.
- **executed**: [events per second] executed events per second.

API

The API thread is processing grapql queries.

- **executed**: [queries per second] executed queries per second.

STREAMING

The STREAMING thread is generating unified events.

- **events**: [events per second] processed events per second.

COMMLIB WORKER THREADS(WPT), COMMLIB READ(RT) and COMMLIB WRITE(WT) THREADS

The logged thread names contain the strings "wpt", "rt" and "wt", e.g. "qmaster(wpt-01)". The threads will show the following information:

- mode **rw**: [actions per second] nr of read or/and a write actions done
- mode **r**: [actions per second] nr of read actions done
- mode **w**: [actions per second] nr of write actions done

- cs **d**: [actions per second] nr of actions done for connections in “disconnected” state
- cs **o**: [actions per second] nr of actions done for connections in “opening” state
- cs **a**: [actions per second] nr of actions done for connections in “accepting” state
- cs **cg**: [actions per second] nr of actions done for connections in “connecting” state
- cs **cd**: [actions per second] nr of actions done for connections in “connected” state
- cs **cl**: [actions per second] nr of actions done for connections in “closing” state

COMMLIB UPDATE THREAD(UT)

This threads contain the string “ut”.

The threads will show the following information:

- hostlist **hletc**: [elements per second] nr of handled host list elements
- hostlist **hlerc**: [elements per second] nr of resolved host list elements
- endplist **eletc**: [elements per second] nr of handled endpoint list elements
- resolving **h2ip4**: [elements per second] nr of hosts resolved via IPv4 address hostname parsing
- resolving **os**: [elements per second] nr of hosts resolved by system call
- resolving **err**: [elements per second] nr of hostname resolving errors

COMMLIB SERVICE THREADS(ST)

This thread contains the string “st”.

The thread will show the following information:

- stat **clestc**: [elements per second] nr of handled elements for statistical data
- debug **dcm**: [elements per second] nr of handled debug clients
- timechk **dcm**: [elements per second] nr of handled messages when doing timeout check

DEADLOCK DETECTION

If there is the assumption that one or more qmaster threads run into a deadlock, it is possible to enable the deadlock detection feature. If enabled this feature will log detected problems into the qmaster messages file. There will be an additional thread started which is checking the lock requests of the registered threads. Once a thread does not get the requested lock for the configured “dlock_timeout” time the deadlock thread will print out all available information. This can help development to find the root cause of the problem.

Since every thread has to update their lock states this feature will cost some performance. There are currently two deadlock detection modes implemented for the qmaster daemon: The “full” mode will register all important threads, including threads doing communication. If a deadlock problem is not affecting the communication library, it is recommended to use the “default” mode to get less performance loss.

The deadlock detection cannot be enabled at runtime. In order to enable it, the bootstrap file must be modified (see *sge_bootstrap*(5) man page). Once modified the qmaster must be restarted.

The timeout parameter “dlock_timeout” is configurable at runtime in the “qmaster_params” line with qconf -mconf (see *sge_conf*(5) man page). If the configured timeout is to low, the deadlock thread might report a deadlock to early (false positive logging). This can happen if an action that is executed while a thread is holding a lock is taking longer than the specified “dlock_timeout” timeout. Usually a lock should not be held for a long time so the default timeout of 540 seconds should be ok. If there are to many false positive loggings the timeout might be set to some higher value.

It is also possible to force an abort() of the sge_qmaster daemon by setting the “qmaster_params” parameter “dlock_abort” (see *sge_conf*(5) man page).

DIRECTORY SERVICES

USER AND GROUP QUERIES

It is possible to change the default timeout used for queries to *group* and *user database* of the operating system. If none of the following methods is used, the default timeout is set to 1 second. The max. allowed setting for this kind of queries is 10 seconds.

1. **Specify timeout via environment variable setting**: It is possible to set the environment variable **SGE_DIRECTORY_SERVICE_TIMEOUT** to a value between 1-10. This will be the used timeout in seconds.
2. **Specify timeout via execd_params**: The execd_params **ENABLE_DIR_SERVICE_TIMEOUT** can also be used to set this timeout when jobs are started at execution daemon (see *sge_conf*(5) man page).

HOST RESOLVING QUERIES

The environment variable **SGE_RESOLVE_SETTING** can be used to influence the way, how queries for resolving host names and ip addresses are done. It is possible to set 3 parameters (with delimiter=“:”, e.g. “SGE_RESOLVE_SETTING=4:16:2000000”) which would have an effect to the Altair Grid Engine host resolving behavior:

1. Parameter: **Specify the used resolve method**: Supported value 1-4 with following meaning:

Value	Description
1	Never retry resolving on error.
2	As long as the service returns **TRY_AGAIN** repeat resolving.
3	Repeat resolving on **TRY_AGAIN** response from the server but stop when timeout occurs.
4	On any error retry again until timeout occurs. This should be used only for debugging.

2. Parameter: **Repeat timeout in Seconds**: This parameter is used to specify the timeout for retries for resolve method 3+4. The timeout unity is **seconds**.
3. Parameter: **Waiting time for resolve retry**: This parameter is used to specify the minimum waiting time after a faulty query, before a retry query is started. The timeout unity is **microseconds**.

If no environment variable is set, the default resolve method is method nr. 3 with 60 seconds timeout and a waiting time of 2000000 (2 seconds).

Example:

```
SGE_RESOLVE_SETTING=3:120:100000 ; export SGE_RESOLVE_SETTING
```

This will repeat resolving, if service answers with **TRY_AGAIN**. If there is still no successful response after 120 seconds, the resolving will fail with an error. The retry frequency will be max. 10 times per second, since the waiting time after a not successful query is set to 100000 microseconds.

QPING MONITORING

The qping(1) command provides monitoring output of Altair Grid Engine components.

REQUEST QUEUES

Requests that are accepted by qmaster but that cannot be immediately handled by one of the reader or worker threads are stored in qmaster internal request queues. qping(1) is able to show details about those pending requests when this is enabled by defining the parameter *MONITOR_REQUEST_QUEUES* as *qmaster_param* in the global configuration of Altair Grid Engine. The output format of requests is the same as for requests log messages (explained in the section *Logging -> Request execution* above).

SCHEDULER

SCHEDULER PROFILING

The scheduler profiling can be switched on in order to find out the root cause of long scheduling times. Monitoring the scheduler profiling data over a long period can also help to figure out the impact of configuration changes on the scheduler performance.

OVERVIEW DEFAULT PROFILING

Scheduler profiling can be switched on by adding the value “PROFILE=true” to the “params” parameter at the scheduler configuration (qconf -msconf). Once enabled the scheduler thread will add profiling information for each scheduler activity to the sge_qmaster messages file. Profiling can be turned off again by just setting “PROFILE=false” or by removing the “PROFILE” keyword from the “params” parameter. There is no need for restarting the scheduler thread.

The profiling messages in the sge_qmaster messages will follow this syntax:

```
<date> <time>|<thread name>|<hostname>|<message-type>|[PROF:|PROF(<thread
id>):] <profiling-message>
```

Value	Description
date	MM/DD/YYYY (date of logging)
time	HH:MM:SS.mmm (time of logging with milliseconds)
thread name	scheduler000 (name of thread which is logging the message)
hostname	hostname where the component is running
message-type	P (Profiling)
thread id	system internal thread id number
profiling-message	Any profiling related output

Profiling types

There are basically two different types of profiling-messages logged:

1. Sequence Specific Logging

A sequence specific profiling output will always start with "PROF: " prefix and is logged when the scheduler thread is reaching some code where profiling output is created. This helps to understand the logical order of the scheduler run.

2. Profiling Summary

The profiling summary is printed at the end of the scheduler loop if scheduler got new events to handle. It provides information for the complete recent scheduler run. This kind of messages will show the prefix "PROF(<thread id>): ". The <thread id> is usually some integer value (e.g. "PROF(205510400): "). The profiling summary is shown as a block of multiple lines.

Sequence Specific Logging description

“job ticket shares and usage calculation” and “running jobs ticket calculation” message

```
Example:
...|PROF: job ticket shares and usage calculation took 0.001 s (init: 0.000, pass 0: 0.001, pass
...PROF: running jobs ticket calculation took 0.000 s (init: 0.000, pass 0: 0.000, pass 1: 0.000,
```

The messages show the wallclock time required for different steps of the ticket calculation and the number of (array) tasks handled by the ticket calculation algorithm. Task counts are capped at the number of free slots + 1 per job, therefore the number of tasks reported here can significantly differ from the actual number of running and pending tasks in the cluster.

Step	Description
init	Initialize data structures
pass 0	Decay past usage, calculate the targeted proportional share for each sharetree node
pass 1	sum up job shares in sharetree nodes, sum up functional user, project, department, job shares
pass 2	get weighting factors, calculate sharetree tickets, calculate functional tickets and apply weighting factors (weight_user, weight_project, ...) apply override tickets
calc	apply the policy hierarchy, combine tickets into total tickets, calculate ticket percentage, update share tree nodes for finished jobs, copy tickets from internal data structures to job/ja_task, cleanup internal data structures
tasks	total number of array tasks handled by the ticket calculation
running	number of running tasks of all jobs (capped, see above)
pending	number of pending tasks of all jobs (capped, see above)
unenrolled	number of unenrolled tasks of all jobs (capped, see above)

"job dispatching took ..." message

```
Example:
...|PROF: job dispatching took 0.000 s (1 fast, 0 fast_soft, 0 pe, 0 pe_soft, 0 res, util 0.000%)
```

The first time shows the wallclock time needed for dispatching all jobs. This measurement is the sum of the times spend in "job dispatching", "rqs parallel" and "rqs sequential". It may also include some fraction of time measured for the "send orders" level if the scheduler decides to send orders while scheduling. If the scheduler sends orders while dispatching it will leave the current level and switch over to the "send orders" layer. This is an exceptional case only done for the "send orders" layer.

Additional info	Description
fast	Nr of sequential jobs calculations
fast_soft	Nr of sequential jobs calculations that had a -soft request
pe	Nr of parallel job calculations
pe_soft	Nr of parallel job calculations that had a -soft request
res	Nr of reservations done
util	utilization during job dispatching

"parallel matching ...", "sequential matching ..." messages

```
Example:
...|PROF: matching counters:        global          rqs     cqstatic      hstatic      qstatic
...|PROF: parallel matching              0            0            0            0            0
...|PROF: sequential matching            1            0            1            1            1
```

The profiling lines for parallel and sequential matching always show seven numbers. These numbers represent the value of internal counters that are increased by one for each matching operation. The counters are placed in different areas of the scheduler code. In this example the job was scheduled with a minimal matching count of "1". This means only one matching operation was needed in the various areas to handle the job. Since there was no resource quota set active there was also no matching operation counted in the "rqs" variable. If the values are increasing compared to older runs this would mean that the scheduler overhead has increased. The numbers represent the areas "global", "rqs", "cqstatic", "hstatic", "qstatic", "hdyn" and "qdyn"

Additional info	Description
global	resources that are defined at global host
rqs	resource quota set calculations
cqstatic	cluster queue static (resources with static value defined at cluster queue level)
hstatic	host static (resources with static value defined at host level)
qstatic	queue static (resources with static value defined at queue level)
hdyn	host dynamic (resources with dynamic value defined at host level)
qdyn	queue dynamic (resources with dynamic value defined at queue level)

"scheduled in ..." message

```
Example:
...|PROF: scheduled in 0.058 (u 0.001 + s 0.002 = 0.003): 1 sequential, 0 parallel, 7 orders, 5 H
```

The first time shows the wallclock time needed for the "scheduling" level. It includes the measurements for all its sub-levels which are "ssos init", "job dispatching", "rqs parallel", "rqs sequential", "priority calculation", "job sorting", "ticket calculation" and "send orders". The parenthesized text shows the user time, system time and the sum of user + system time.

Additional info	Description
sequential	Nr of scheduled (started) sequential jobs
parallel	Nr of scheduled (started) parallel jobs
orders	Nr of orders created for the worker threads
H	Nr of hosts configured
Q	Nr of currently available queue instances

Additional info	Description
QA	Nr of queue instances configured
J(qw)	Nr jobs in pending state
J(r)	Nr jobs in running state
J(s)	Nr jobs in suspended state
J(h)	Nr jobs in hold state
J(e)	Nr jobs in error state
J(x)	Nr jobs in finished state
J(all)	Total job count
C	Nr of configured complexes
ACL	Nr of defined user sets (user sets with type ACL)
PE	Nr of defined pes
U	Nr of defined users
D	Nr of defined departments (user sets with type DEPT)
PRJ	Nr of defined projects
STN	Nr of share tree nodes
STL	Nr of share tree leafs
CKPT	Nr of defined checkpointing environments
gMes	Nr of “global” scheduler messages (if schedd_info is turned on)
jMes	Nr of “job” related messages (if schedd_info is turned on)
pre-send	Shows the total number of orders and the nr of order packages that was created while scheduling. If “7/4” is logged this would mean the scheduler has send 4 packages for 7 orders so far.

“schedd run took . . .” message

```
Example:
...|PROF: schedd run took: 0.080 s (init: 0.000 s, copy: 0.000 s, run: 0.061, free: 0.000 s, jobs
```

This profiling message is printed at the end of the scheduler run and shows the complete wallclock time (including sub-levels) for the the last scheduling run. At this point all orders that were sent by the scheduler have been handled by the qmaster worker threads.

Additional info	Description
init	Wallclock time needed for initialization of data structures (including sub-level measurement)
copy	Wallclock time needed for copy actions (including sub-level measurement)
run	Wallclock time needed for the pure scheduler calculations (including sub-level measurement)
free	Wallclock time needed for cleanup (including sub-level measurement)
jobs	Total nr of jobs in the system
categories	Nr of job categories referenced in the job / Nr of job categories in JC_FILTER list (if scheduler params JC_FILTER is set)
run_util	The utilization for the pure scheduler calculations

Additional info	Description
total_util	The utilization for the complete last scheduler run

Profiling Summary description

The profiling summary is logged into the qmaster messages file after a full scheduler run. Each line of the profiling summary shows the wallclock (wc), the user time (utime), the system time (stime) and cpu utilization for each profiling area (level). The cpu utilization is the ratio of wallclock time compared to cpu time (system + user time) for the specific level. If the scheduler thread shows e.g. utilization of about 100% the scheduler used one complete cpu for calculation.

NOTE: Since Altair Grid Engine versions 8.3.1p7 the scheduler profiling code will provide user and system time on a thread level on Linux kernels > 2.6.26 and Solaris operating systems.

The values for utime and stime are only correct if "LWP based" (LWP = Light Weight Process) is shown in the profiling output. If you see "process based" or "times based" there then only the wallclock time is shown correctly. The shown text depends on the system architecture and should be "LWP based" for newer Linux and Solaris operating systems. Systems that do not support per thread specific usage calls will always show the system and user time of the complete process which also includes the system and user times of other threads and therefore only the wallclock times can be used for further analysis.

Profiling type	Description
LWP based	Thread based measurement is supported. The system and user times show the calling thread values. (LWP=Light Weight Process)
process based	Process based measurement is supported. The system and user times show the entire process values. The `getrusage()` system call does **not** support threads.
times based	The `times()` system call is used to obtain only wallclock time. The system and user times are not measured at all.

The profiling areas show the hierarchical order of the scheduler code. It has following hierarchy tree:

```
*--"scheduler thread"
   |
   *--"waiting for events"
   |  |
   |  *--"mirror events"
   |
   *--"scheduler event loop"
   |  |
   |  *--"config update"
   |  |
   |  *--"data preparation"
```

```
   |  |  |
   |  |  *--"scheduling"
   |  |     |
   |  |     *--"ssos init"
   |  |     |
   |  |     *--"job dispatching"
   |  |     |  |
   |  |     |  *--"rqs parallel"
   |  |     |  |
   |  |     |  *--"rqs sequential"
   |  |     |
   |  |     *--"priority calculation"
   |  |     |  |
   |  |     |  *--"job sorting"
   |  |     |  |
   |  |     |  *--"ticket calculation"
   |  |     |
   |  |     *--"send orders"
   |  |
   |  *--"wait for order completion"
   |
   *--"set event client params"

*--"other"
*--"total"
```

The parent levels will not show usage values accounted in their sub-levels. In order to get the times spend in e.g. “job dispatching” the sum of the times printed in sub-level “rqs parallel” and sub-level “rqs sequential” must be added to the values of “job dispatching”.

Profiling level	Description
“scheduler thread”	This layer covers the complete scheduler thread main loop.
“waiting for events”	Time spend waiting for arrival of new events. The events contain updates for the scheduler database.
“mirror events”	Time needed for updating schedulers database.
“scheduler event loop”	The scheduler thread enters this layer once a new scheduler run must be started.
“config update”	If there was a configuration change the code executing the update will be profiled in this level.
“data preparation”	There are various areas in the code where some data has to be prepared (copied, lists initialized, etc.). All this time is measured in this level.
“scheduling”	The scheduling level is the starting level of Altair Grid Engine scheduling code.
“ssos init”	If slotwise suspend on subordinate is configured the initialization for this is profiled in this level.
“job dispatching”	This level is used to profile the job scheduling code.

Profiling level	Description
“rqs parallel”	If resource quotas are in use the profiling for parallel jobs is there.
“rqs sequential”	If resource quotas are in use the profiling for sequential jobs is there.
“priority calculation”	All job order calculations are measured in this level.
“job sorting”	Pure job sorting operations level.
“ticket calculation”	Adjusting resulting tickets for each job is profiled here.
“send orders”	Measures the time spent for sending orders to the qmaster worker threads.
“wait for order completion”	This is a synchronization point for the scheduler and the worker threads. The scheduler has to wait until the workers have processed all orders sent by the scheduler for this scheduler run.
“set event client params”	All calls to the event API (changes of configuration, event busy handling, ...) is measured here.
“other”	All not covered cpu activity should show up here. The values should be zero for the scheduler thread.
“total”	The last line shows the sum of all layers. Since Altair Grid Engine 8.6.8 each profiling level line also contains the proportion of the level specific wall clock time compared to the total wall clock time (wc total = ...%)

RQS PROFILING

In order to get a feeling of how resource quota settings influence the scheduling performance there is an additional profiling output printed. If resource quotas are defined the profiling output will show additional information for each resource quota rule that takes longer than 0.001 seconds. Since the scheduler does different calculations for sequential and parallel jobs the RQS rule is shown for each of the job types:

```
SGE/master> qconf -srqs RQS_0
{
   name         RQS_0
   description  NONE
   enabled      TRUE
   limit        name hgrp_limit hosts @HGRP_0 to slots=90
}

Example:
...|PROF: sequential RQS "RQS_0"(rule nr. 1) took 1.996 s (1800 calculations, 0 calc_cached, 1800
...|PROF: sequential RQS "RQS_0"(no rule match) took 0.010 s (0 calculations, 0 calc_cached, 0 vi
...|PROF: parallel RQS "RQS_0"(rule nr. 1) took 0.184 s (200 calculations, 44 calc_cached, 154 vi
```

```
...|PROF: parallel RQS "RQS_0"(no rule match) took 0.002 s (0 calculations, 0 calc_cached, 0 viol
```

The message contains the RQS name (RQS_0) and which rule nr (rule nr. 1) was measured. If the string “no rule match” is shown the time needed for figuring out that the rule does not influence the current job dispatching is measured. The resource quota sets are sorted alphabetically in order to have a fix verification order. Quotas that are limiting more resources should be handled first in order to optimize the calculation time. It does not make sense to check first the rules that will not have any affect most of the time.

Additional info	Description
calculations	Nr of quota calculations that have to be done for this rule
calc_cached	Nr of times some calculation could use a cached value
violations	Nr of detected violations for this rule

JOB SPECIFIC PROFILING

An additional job specific scheduler profiling can be switched on by setting “WARN_DISPATCHING_TIME” to a non zero value in the “params” parameter at the scheduler configuration. The WARN_DISPATCHING_TIME was introduced to print warning messages for each job taking longer than the defined value in milliseconds to dispatch. If the time is set to a high value (e.g. 10000 ms) the warning messages will not appear but the scheduler profiling will still show additional information for the jobs with the fastest (top job) and with the slowest (low job) dispatch time per scheduling run:

```
Example:
...|PROF: min. dispatching time was 0.064s for job 3000001063.1 (reservation=false)
...|PROF: max. dispatching time was 0.095s for job 3000001907.1 (reservation=false)
```

Minimum and maximum job dispatching time is printed out. The logging shows also if this calculation was done for reserving the job.

Top job and low job information messages

The “job profiling” lines show detailed information for the scheduling dispatch calculation of the job with the printed job id. The times for initialization, assignment, debiting and cleanup of the job calculation and many additional values which are explained in the table below.

```
Example:
...|PROF: job profiling(top job) of 3000001063.1 - category: -l mem_free=62,swap_total=100,swap_u
...|PROF: job profiling(top job) of 3000001063.1 - CATINT=-1 CATREF=1
...|PROF: job profiling(top job) of 3000001063.1 - init of job took 0.000007s
...|PROF: job profiling(top job) of 3000001063.1 - assignment of job took 0.064668s
...|PROF: job profiling(top job) of 3000001063.1 - debiting of job took 0.000002s
...|PROF: job profiling(top job) of 3000001063.1 - cleanup of job took 0.000010s
...|PROF: job profiling(top job) of 3000001063.1 - RESULT=-1 PE=1 START=1 RES=0
...|PROF: job profiling(top job) of 3000001063.1 - SAC=0 PAC=1
...|PROF: job profiling(top job) of 3000001063.1 - PESL=1 PEAC=4 MINS=2 MAXS=23
```

```
...|PROF: job profiling(top job) of 3000001063.1 - LSH=1 LMSH=0 PEALG=2 PEROS=-2
...|PROF: job profiling(top job) of 3000001063.1 - PETQL1=4 PETQL2=4 PETQL3=4 PETQL4=2020 PETQL5=
...|
...|PROF: job profiling(low job) of 3000001907.1 - category: -l mem_free=904,swap_total=100,swap_
...|PROF: job profiling(low job) of 3000001907.1 - CATINT=-1 CATREF=1
...|PROF: job profiling(low job) of 3000001907.1 - init of job took 0.000019s
...|PROF: job profiling(low job) of 3000001907.1 - assignment of job took 0.095002s
...|PROF: job profiling(low job) of 3000001907.1 - debiting of job took 0.000056s
...|PROF: job profiling(low job) of 3000001907.1 - cleanup of job took 0.000001s
...|PROF: job profiling(low job) of 3000001907.1 - RESULT=-1 PE=1 START=1 RES=0
...|PROF: job profiling(low job) of 3000001907.1 - SAC=0 PAC=1
...|PROF: job profiling(low job) of 3000001907.1 - PESL=1 PEAC=4 MINS=2 MAXS=23
...|PROF: job profiling(low job) of 3000001907.1 - LSH=1 LMSH=0 PEALG=2 PEROS=-2
...|PROF: job profiling(low job) of 3000001907.1 - PETQL1=4 PETQL2=4 PETQL3=4 PETQL4=2020 PETQL5=
```

Additional info	Description
category	The category string created for the job. The category string shows the requested resources
CATINT	Number of categories used in this scheduling or “-1”. “-1” tells that only CATREF is used
CATREF	Number of jobs referencing this category
RESULT	The job dispatch result: 4: An error happened, no dispatch will ever work again for this job 2: Attribute does not exist error 1: No assignment possible at the specified time 0: Job dispatched -1: Assignment will never be possible for any job of that category -2: Assignment will never be possible for that particular job
PE	This value is set to “1” for jobs requesting a PE and “0” for sequential jobs
START	This value is set to “1” for jobs that should be dispatched right now. It is set to “0” for e.g. reservations or jobs with a later start time.
RES	This value is set to “1” if it is a reservation calculation
SAC	Nr of serial assignment calculations for this job
PAC	Nr of parallel assignment calculations for this job
PESL	Nr of needed PE Search Loops (> 1 if more than one PE is matching)
PEAC	Nr of needed PE assignment calculations for the PE optimize slot search algorithm)
MINS	Nr of min. requested PE slots
MAXS	Nr of max. requested PE slots
LSH	Nr of hosts in local skip host list for PE task assignment search

Additional info	Description
LMSH	Nr of hosts in local skip master host list for PE task assignment search
PEALG	The internal number of the PE optimize slot search algorithm -1: Automatic, the scheduler will decide 0: Least slot first 1: Highest slot first 2: Binary search
PEROS	Internal number of allocation rule or slots per host >0: Allocate exactly this number on each host 0: Unknown allocation rule -1: Indicates that hosts should be filled up sequentially -2: Indicates that a round robin algorithm with all available host is used
PETQL1-PETQL5	Counters for parallel slot allocation nested loops. High values indicate expensive calculation effort.

JOB SPECIFIC TRACING

There are the three **qmaster_params** attributes **JOB_SPECIFIC_TRACING_TARGET**, **JOB_SPECIFIC_TRACING_DEBUG_LEVEL**, **JOB_SPECIFIC_TRACING_ID** which allow to define a job for which both debug tracing and scheduler runlog writing shall be automatically enabled and be disabled after this job was scheduled.

This is useful in huge clusters where it is not possible to have the debug tracing or the scheduler runlog enabled all the time because of its performance impact, if the scheduling of certain jobs shows faulty behaviour.

Specify e.g.:

```
params   JOB_SPECIFIC_TRACING_TARGET=/tmp/trace.log, \
         JOB_SPECIFIC_TRACING_DEBUG_LEVEL=3:0:0:0:0:0:0:0, \
         JOB_SPECIFIC_TRACING_ID=3000000036
```

to get the debug tracing of the Scheduler run for the single job 3000000036 in the file "/tmp/trace.log". The scheduler runfile is written to its default location at the same time, i.e. to "$SGE_ROOT/$SGE_CELL/common/schedd_runlog".

Both could be useful for the Altair Grid Engine support people to identify problems in the Scheduler code regarding these specific job types.

Attention, both the target debug trace file and the scheduler runlog are overwritten every time the scheduler starts to schedule the given job, which means they are overwritten not only with every scheduler run, but could also be overwritten within one scheduler run if the job is an array job where different array tasks are scheduled in different step.

These three settings params are meant for very specific debugging purposes, they overwrite any debug tracing levels or targets defined in the environment or the **qmaster_params** and do not restore the old settings after the specified job has been scheduled!

See the man page sge_conf(5) for more details regarding these three **qmaster_params** attributes.

RESOURCE DIAGRAM

Scheduler thread has so called resource diagrams which contain the information which resources are in use over time.

Resource diagrams are used when resource reservation is enabled (sge_sched_conf(5) max_reservation > 0) or when advance reservations exist in the cluster.

They can be dumped to file by setting WRITE_RESOURCE_DIAGRAM=TRUE or WRITE_RESOURCE_DIAGRAM=APPE in the params attribute of the scheduler configuration. When WRITE_RESOURCE_DIAGRAM=TRUE is set and qconf -tsm is called, the resource diagram in place after the scheduling run is dumped to $SGE_ROOT/$SGE_CELL/common/resource_diagram. If WRITE_RESOURCE_DIAGRAM=APPEND is set, the resource diagram is appended to the file. Otherwise the content is overwritten.

Resource diagrams are created for every consumable complex variable which has a capacity on global, exec host or queue instance level, as well as for slots in parallel environments.

Please note that the dump of resource diagrams on queue instance level can be incomplete.

Example for resource diagrams with two jobs requesting a global consumable “lic”, one of the jobs is a parallel job requesting 2 slots in the pe “mytestpe”, slots are defined on exec host level:

```
PARALLEL ENVIRONMENT "mytestpe":
resource utilization: mytestpe "slots" 2.000000 utilized now
        03/07/2018 10:21:21  2.000000
        03/07/2018 10:25:41  0.000000
resource utilization: mytestpe "slots" 0.000000 utilized now non-exclusive
------------------------------------------
GLOBAL HOST RESOURCES
resource utilization: global "lic" 3.000000 utilized now
        03/07/2018 10:20:31  1.000000
        03/07/2018 10:21:21  3.000000
        03/07/2018 10:24:51  2.000000
        03/07/2018 10:25:41  0.000000
resource utilization: global "lic" 0.000000 utilized now non-exclusive
[...]
EXEC HOST "rgbfs"
resource utilization: rgbfs "slots" 2.000000 utilized now
        03/07/2018 10:21:21  2.000000
        03/07/2018 10:25:41  0.000000
resource utilization: rgbfs "slots" 0.000000 utilized now non-exclusive
[...]
EXEC HOST "lauchert"
resource utilization: lauchert "m_mem_free" 0.000000 utilized now
resource utilization: lauchert "m_mem_free" 0.000000 utilized now non-exclusive
resource utilization: lauchert "slots" 1.000000 utilized now
        03/07/2018 10:20:31  1.000000
```

```
        03/07/2018 10:24:51  0.000000
resource utilization: lauchert "slots" 0.000000 utilized now non-exclusive
[...]
```

GRID ENGINE ERROR, FAILURE AND EXIT CODES

Altair Grid Engine provides a number of job or feature related exit codes, which can be used to trigger a job or a queue behaviour and a resulting consequence, for either the job or also the queue. These exit codes are shown in the following tables.

Job related error and exit codes

The following table lists the consequences of different job-related error codes or exit codes. These codes are valid for every type of job.

Script/Method	Exit or Error Code	Consequence
Job Scrips	0	Success
	99	Re-queue
	Rest	Success: Exit code in accounting
Epilog/Prolog	0	Success
	99	Re-queue
	100	Job in Error state
	Rest	Queue in Error state, Job re-queued

Parallel-Environment-Related Error or Exit Codes

The following table lists the consequences of error codes or exit codes of jobs related to parallel environment (PE) configuration.

Script/Method	Error or Exit Code	Consequence
pe_start	0	Success
	Rest	Queue set to error state, job re-queued
pe_stop	0	Success
	Rest	Queue set to error state, job not re-queued

Queue-Related Error or Exit Codes

The following table lists the consequences of error codes or exit codes of jobs related to queue configuration. These codes are valid only if corresponding methods were overwritten.

Script/Method	Error or Exit Code	Consequence
Job Starter	0	Success
	Rest	Success, no other special meaning
Suspend	0	Success
	Rest	Success, no other special meaning
Resume	0	Success
	Rest	Success, no other special meaning
Terminate	0	Success
	Rest	Success, no other special meaning

Checkpointing-Related Error or Exit Codes

The following table lists the consequences of error or exit codes of jobs related to checkpointing.

Script/Method	Error or Exit Code	Consequence
Checkpoint	0	Success
	Rest	Success. For kernel checkpoint, however, this means that the checkpoint was not successful.
Migrate	0	Success
	Rest	Success. For kernel checkpoint, however, this means that the checkpoint was not successful. Migration will occur.
Restart	0	Success
	Rest	Success, no other special meaning
Clean	0	Success
	Rest	Success, no other special meaning

qacct -j “failed” line Codes

For jobs that run successfully, the qacct -j command output shows a value of 0 in the failed field, and the output shows the exit status of the job in the exit_status field. However, the shepherd might not be able to run a job successfully. For example, the epilog script might fail, or the shepherd might not be able to start the job. In such cases, the failed field displays one of the code values listed in the following table.

Code	Description	qacct	Meaning for Job
0	No failure	valid	Job ran, exited normally
1	Presumably before job	invalid	Job could not be started
3	Before writing config	invalid	Job could not be started

Code	Description	qacct	Meaning for Job
4	Before writing PID	invalid	Job could not be started
5	On reading config file	invalid	Job could not be started
6	Setting processor set	invalid	Job could not be started
7	Before prolog	invalid	Job could not be started
8	In prolog	invalid	Job could not be started
9	Before pestart	invalid	Job could not be started
10	In pestart	invalid	Job could not be started
11	Before job	invalid	Job could not be started
12	Before pestop	valid	Job ran, failed before calling PE stop procedure
13	In pestop	valid	Job ran, PE stop procedure failed
14	Before epilog	valid	Job ran, failed before calling epilog script
15	In epilog	valid	Job ran, failed in epilog script
16	Releasing processor set	valid	Job ran, processor set could not be released
17	Through signal	valid	Job ran, shepherd got signal
18	Shepherd returned error	valid	Job ran, shepherd returned an error
19	Before writing exit_status	valid	Job started, failed before writing error status
20	Found unexpected error file	valid	Job ran, shepherd found non-empty error file
21	In recognizing job	valid	Execd tells qmaster that it doesn’t know about this job
22	Job lost at execution host	valid	Job ran, execd stopped reporting, job rescheduled
23	Can’t get PIDs	valid	Currently not in use.

Code	Description	qacct	Meaning for Job
24	Migrating (check-pointing jobs)	valid	Job ran, job will be migrated
25	Rescheduling	valid	Job ran, job will be rescheduled
26	Opening output file	invalid	Job could not be started, stderr/stdout file could not be opened
27	Searching re-quested shell	invalid	Job could not be started, shell not found
28	Changing to working directory	invalid	Job could not be started, error changing to start directory
29	No message -> AFS problem	invalid	Job could not be started
30	Rescheduling on appli-cation error	invalid	Job ran until application failed, rescheduling
31	Accessing sgepasswd file	invalid	Job could not be started, job failure
32	Entry is missing in password file	invalid	Job could not be started, job failure
33	Wrong password	invalid	Job could not be started, job failure
34	Communicating with Grid Engine Helper Service	invalid	Job could not be started, job failure
35	Before job in Grid Engine Helper Service	invalid	Job could not be started, job failure

Code	Description	qacct	Meaning for Job
36	Checking configured daemons	invalid	Job could not be started, job failure
37	Qmaster enforced h_rt limit	valid	Job was killed by qmaster h_rt limit (if qmaster_params ENABLE_ENFORCE_MASTER_LIMIT is set).
38	Additional group id could not be set	invalid	Job could not be started, ADD_GRP_ID can not be set
39	Before pe task prolog	valid	Job pe task could not be started, pe prolog failure
40	In pe task prolog	valid	Job pe task could not be started, pe prolog script failure
41	Before pe task epilog	valid	Job pe task ran, pe epilog failure
42	In pe task epilog	valid	Job pe task run, pe epilog script failure
43	Obtaining password failed with out of memory error	valid	Execd runs out of memory when obtaining user password
44	Execd enforced h_rt limit	valid	Job or pe task was killed by execd because it reached hard runtime limit
45	Execd enforced h_cpu limit	valid	Job or pe task was killed by execd because it reached hard cpu limit, execd_params ENFORCE_LIMITS=EXECD must be set to get this failure code
46	Execd enforced h_vmem limit	valid	Job or pe task was killed by execd because it reached hard vmem limit, execd_params ENFORCE_LIMITS=EXECD must be set to get this failure code
47	Execd enforced h_rss limit	valid	Job or pe task was killed by execd because it reached hard rss limit, execd_params ENFORCE_LIMITS=EXECD must be set to get this failure code

Code	Description	qacct	Meaning for Job
48	Forced job deletion	invalid	Job was killed by a qdel -f (forced job deletion). The accounting record contains all available online usage so far. The exit_status is undefined since the force option will not wait for final job finish data from execution daemon. The “deleted_by” information contains the user name who deleted the job.
49	qmaster enforced reschedule-limit	invalid	Job was rescheduled too many times, thus qmaster enforced the set reschedule-limit, instead of scheduling the job.
50	Docker job error	invalid	A job could not be started in a Docker container because the Docker daemon reported the job arguments are invalid.
51	could not be started due to cgroups-error	invalid	Job could not be started because of an error when setting up the necessary cgroups. E.g. this can happen if the cgroups_path or devices are configured wrongly. More information can be found using **qstat -j** or inside the shepherd trace file.
52	cgroups enforced memory limit	valid	Job exceeded its requested memory limit and was killed by the cgroup OOM-killer.
53	Docker job communi-cation error	invalid	The communication between the Altair Grid Engine components running on the host and in the container failed.
54	GPU config error	?	Currently not in use.
55	epilog cgroups error	?	If ignore_epilog_cgroups_error=0 the queue is set to error state.
56	pe_stop cgroups error	?	If ignore_epilog_cgroups_error=0 the queue is set to error state.
57	Docker container could not be started	invalid	The container for the job or task did not start, so the job/task also could not be started, job failure.
100	Assumedly after job	valid	Job ran, job killed by a signal or other failure

Code	Description	qacct	Meaning for Job
101	Parallel job finished with missing slave task report	invalid	Job ran, qmaster_params ENABLE_JOB_FAILURE_IF_SLAVE_TASK_MISSING must be enabled to get this failure code
102	Parallel job finished with at least one non-zero slave task exit status	valid	Job ran, qmaster_params ENABLE_JOB_FAILURE_ON_SLAVE_TASK_ERROR must be enabled to get this failure code
103	Parallel job finished with at least one slave task reporting failed state	valid	Job ran, qmaster_params ENABLE_JOB_FAILURE_ON_SLAVE_TASK_ERROR must be enabled to get this failure code

The Code column lists the value of the failed field. The Description column lists the text that appears in the qacct -j output. The qacct column shows if the accounting record is valid or invalid. The Meaning for Job column indicates whether the job ran or not.

GRID ENGINE QUEUE AND JOB STATES

QUEUE OR QUEUE INSTANCE STATES

The following table shows all possible queue or queue instance states. These states will be shown with the qstat command in the *states* column

State	Description	Trigger	Consequence
u	qinstance/host is in unknown state	exec node sends no load reports, due to network issues or execution node/daemon is down	No new job will be transferred to the this node. Running jobs will stay running as long as the node is working

State	Description	Trigger	Consequence
a	qinstance is in load alarm state	load_threshold is exceeded (see. cluster queue configuration)	Running jobs will stay running, no new job will be dispatched to this queue instance, to prevent host from being overloaded
s	qinstance is in suspended state (manual suspension)	manual suspension of queue instance (qmod -s	Running jobs will be suspended, no new job will be dispatched to this queue or queue instance
d	qinstance is in disabled state (manual disabling)	manual disabling of queue instance (qmod -d	Running job will stay running, no new job will be dispatched to this queue or queue instance
c	qinstance is in configuration ambiguous state	queue configuration conflict found	Running jobs will stay running, no new job will be dispatched to this queue or queue instance
o	qinstance is in orphaned state	queue configuration is deleted, while jobs are still running. will disappear when last running is finished	Running jobs will stay running, no new job will be dispatched to this queue or queue instance
A	qinstance is in suspend alarm state	queue instance exceeds suspend_threshold	Running jobs will be suspended within the suspension interval (sge_queue_conf(5) *nsuspend*) until no suspend_threshold is exceeded anymore no new job will be dispatched to this queue or queue instance

State	Description	Trigger	Consequence
B	qinstance is in blocked state due to SSOS (Slotwise Suspension On Subordinate)	Total number of jobs on this subordinate queue instance and its parent queues exceeds the configured threshold value.	This subordinate queue instance will be blocked from running any new jobs until the total number of jobs on the subordinate and its parents is below the threshold. Jobs may still be run on the parent queue instance.
__	————————	————————	————————————————
__-	————	————	—————

CSP Related Errors

When the Qmaster process is not starting and the following message is presented in the emergency file (located in /tmp):

```
main|master01|E|commlib error: ssl error ([ID=336245134] in module "SSL routines": "ca md too
```

Refer to sge_ca(8) man page, section Notes, Changes in the CSP hashing algorithms, for detailed info.

SEE ALSO

sge_intro(1) sge_qmaster(8) sge_execd(8) qconf(1) qping(1) sge_conf(5)

COPYRIGHT

sge_host_aliases

NAME

host_aliases - Altair Grid Engine host aliases file format

DESCRIPTION

All Altair Grid Engine components use a hostname resolving service provided by the communication library to identify hosts via a unique hostname. The communication library itself references standard UNIX directory services such as DNS, NIS and /etc/hosts to resolve hostnames. In rare cases these standard services cannot be setup cleanly and Altair Grid Engine communication daemons running on different hosts are unable to automatically determine a unique hostname for one or all hosts which can be used on all hosts. In such situations a Altair Grid Engine host aliases file can be used to provide the communication daemons with a private and consistent hostname resolution database.

The communication library is also caching the provided name by the UNIX directory services for at least 10 minutes (default). It is possible to change the re-resolving timeouts with the following sge_qmaster configuration parameters:

- DISABLE_NAME_SERVICE_LOOKUP_CACHE
- NAME_SERVICE_LOOKUP_CACHE_ENTRY_LIFE_TIME
- NAME_SERVICE_LOOKUP_CACHE_ENTRY_UPDATE_TIME
- NAME_SERVICE_LOOKUP_CACHE_ENTRY_RERESOLVE_TIME

The caching specific parameters are described in sge_conf(5) man page.

Changes to the host_aliases file are not immediately active for components that are already running. If the changes result e.g. in a different hostname for an execution daemon the daemon must be restarted. At startup of sge_execd or sge_qmaster the database configuration is verified and adjusted. This is also the case if the resulting hostname of the UNIX directory services have changed. If the name of a sge_qmaster or execd host has changed at the UNIX directory services during runtime the running components should be restarted in order to trigger also a verification of the database. Without restart of such daemons the change will be effective once the cached value of the resolved hostname is renewed and it might result in unexpected behavior if previously used hostnames are not resolvable or not unique anymore.

Adding new entries without restarting of the sge_qmaster daemon is possible if the resulting hostnames are not influencing the cluster configuration. The depending configurations

are host configurations, admin host names, execd host names and submit host names. The sge_qmaster daemon will re-read the host_aliases file during runtime from time to time and add some information into the messages logging file. If it would be necessary to restart the sge_qmaster daemon this will also result in a logging in the sge_qmaster messages file.

If an already used hostname should be changed either in the directory services or in the host_aliases file without restarting of the sge_qmaster the affected host might be removed from the database first. Once all references to the old hostname are removed and all daemons running on that host have been shut down, the hostname can be changed. Once the hostname has been changed the previous name could still be cached in the communication library or in system services like named. Please make sure that the running sge_qmaster resolves the hostname correctly before adding the renamed host again. This verification could be done with the gethostbyname command mentioned below.

Making changes of the behavior of the standard UNIX directory services might also make it necessary to restart affected components.

In order to figure out the resulting name of a host at sge_qmaster the gethostbyname option -all_rr could be used (see hostnameutils(1) man page).

The location for the host aliases file is `<sge_root>/<cell>/common/host_aliases`.

FORMAT

For each host a single line must be provided with a blank, comma or semicolon separated list of hostname aliases. The first alias is defined to be the **unique** hostname which will be used by all Altair Grid Engine components using the hostname aliasing service of the communication library. All names used in a single line must be resolvable via UNIX standard directory services.

SEE ALSO

sge_intro(1)

COPYRIGHT

sge_host_conf

NAME

host_conf - Altair Grid Engine execution host configuration file format

DESCRIPTION

host_conf reflects the format of the template file for the execution host configuration. Via the **-ae** and **-me** options of the qconf(1) command, you can add execution hosts and modify the configuration of any execution host in the cluster. Default execution host entries are added automatically as soon as a sge_execd(8) registers to sge_qmaster(8) for the very first time from a certain host. The qconf(1) **-sel** switch can be used to display a list of execution host being currently configured in your Altair Grid Engine system. Via the **-se** option you can print the execution host configuration of a specified host.

The special hostname “global” can be used to define cluster global characteristics.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

The format of a host_conf file is defined as follows:

hostname

The execution hosts name as defined for host_name in sge_types(1).

load_scaling

A comma separated list of scaling values to be applied to each or part of the load values being reported by the sge_execd(8) on the host and being defined in the complex configuration (see sge_complex(5)). The load scaling factors are intended to level hardware or operating system specific differences between execution hosts.

The syntax of a load factor specification is as follows: First the name of the load value (as defined in the complex) is given and, separated by an equal sign, the load scaling value is provided. No blanks are allowed in between the load_scaling value string.

The parameter **load_scaling** is not meaningful for the definition of the “global” host.

complex_values

complex_values defines quotas for resource attributes managed via this host. Each complex attribute is followed by an "=" sign and the value specification compliant with the complex attribute type (see sge_complex(5)). Quota specifications are separated by commas.

The quotas are related to the resource consumption of all jobs on a host in the case of consumable resources (see sge_complex(5) for details on consumable resources) or they are interpreted on a per job slot basis in the case of non-consumable resources. Consumable resource attributes are commonly used to manage free memory, free disk space, available floating software licenses, or access to specific co-processor devices, while non-consumable attributes usually define distinctive characteristics like type of hardware installed.

For consumable resource attributes an available resource amount is determined by subtracting the current resource consumption of all running jobs on the host from the quota in the **complex_values** list. Jobs can only be dispatched to a host if no resource requests exceed any corresponding resource availability obtained by this scheme. The quota definition in the **complex_values** list is automatically replaced by the current load value reported for this attribute, if load is monitored for this resource and if the reported load value is more stringent than the quota. This effectively avoids oversubscription of resources.

A special configuration is needed for resources with type RSMAP (see sge_resource_map(5)).

Note: Load values replacing the quota specifications may have become more stringent because they have been scaled (see **load_scaling** above) and/or load adjusted (see sge_sched_conf(5)). The -F option of qstat(1) and the load display in the qmon(1) queue control dialog (activated by clicking on a queue icon while the "Shift" key is pressed) provide detailed information on the actual availability of consumable resources and on the origin of the values taken into account currently.

Note also: The resource consumption of running jobs (used for the availability calculation) as well as the resource requests of the jobs waiting to be dispatched either may be derived from explicit user requests during job submission (see the -l option to qsub(1)) or from a "default" value configured for an attribute by the administrator (see sge_complex(5)). The -r option to qstat(1) can be used for retrieving full detail on the actual resource requests of all jobs in the system.

For non-consumable resources Altair Grid Engine simply compares the job's attribute requests with the corresponding specification in **complex_values** taking the relation operator of the complex attribute definition into account (see sge_complex(5)). If the result of the comparison is "true", the host is suitable for the job with respect to the particular attribute. For parallel jobs each job slot to be occupied by a parallel task is meant to provide the same resource attribute value.

Note: Only numeric complex attributes can be defined as consumable resources and hence non-numeric attributes are always handled on a per job slot basis.

The default value for this parameter is NONE, i.e. no administrator defined resource attribute quotas are associated with the host.

load_values

This entry cannot be configured but is only displayed in case of a qconf(1) **-se** command. All load values are displayed as reported by the sge_execd(8) on the host. The load values are enlisted in a comma separated list. Each load value start with its name, followed by an equal sign and the reported value.

processors

Note: Deprecated, may be removed in future release.
This entry cannot be configured but is only displayed in case of a qconf(1) **-se** command. Its value is the number of processors which has been detected by sge_execd(8) on the corresponding host.

usage_scaling

The format is equivalent to **load_scaling** (see above), the only valid attributes to be scaled however are **cpu** for CPU time consumption, **mem** for Memory consumption aggregated over the life-time of jobs and **io** for data transferred via any I/O devices. The default NONE means “no scaling”, i.e. all scaling factors are 1.

user_lists

The **user_lists** parameter contains a space separated list of so called user access lists as described in sge_access_list(5). Each user contained in at least one of the enlisted access lists has access to the host. If the **user_lists** parameter is set to NONE (the default) any user has access being not explicitly excluded via the **xuser_lists** parameter described below. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the host.

xuser_lists

The **xuser_lists** parameter contains a space separated list of so called user access lists as described in sge_access_list(5). Each user contained in at least one of the enlisted access lists is not allowed to access the host. If the **xuser_lists** parameter is set to NONE (the default) any user has access. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the host.

projects

The **projects** parameter contains a space separated list of projects that have access to the host. Any projects not in this list are denied access to the host. If set to NONE (the default), any project has access that is not specifically excluded via the **xprojects** parameter described below. If a project is in both the **projects** and **xprojects** parameters, the project is denied access to the host.

xprojects

The **xprojects** parameter contains a space separated list of projects that are denied access to the host. If set to NONE (the default), no projects are denied access other than those denied access based on the **projects** parameter described above. If a project is in both the **projects** and **xprojects** parameters, the project is denied access to the host.

report_variables

The **report_variables** parameter contains a space separated list of variables that shall be written to the reporting file. The variables listed here will be written to the reporting file when a load report arrives from an execution host.

Default settings can be done in the global host. Host specific settings for report_variables will override settings from the global host.

license_constraints

The **license_constraints** parameter contains a space separated list of constraints that limits the usage of the given licenses. These constraints are inherited from License Orchestrator and cannot be changed locally in Altair Grid Engine. A license constraint shows to which License Manager a license belongs to as also the used and total count. Optionally it can express that licenses are limited to special users and/or to hosts. For more information have a look at the License Orchestrator documentation.

license_oversubscription

The **license_oversubscription** parameter contains a space separated list of licenses whose reported load values are supposed to get increased. Each license attribute is followed by an "=" sign and the count which should get added to the reported load value. The load value only gets increase when the reported value is 0 and not all of this licenses are consumed by this cluster.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), uptime(1), sge_access_list(5), sge_complex(5), sge_execd(8), sge_qmaster(8).

COPYRIGHT

sge_hostgroup

NAME

hostgroup - host group entry file format

DESCRIPTION

A host group entry is used to merge host names to groups. Each host group entry file defines one group. Inside a group definition file you can also reference to groups. These groups are called subgroups. A subgroup is referenced by the sign "@" as first character of the name.

A list of currently configured host group entries can be displayed via the qconf(1) **-shgrpl** option. The contents of each enlisted host group entry can be shown via the **-shgrp** switch. The output follows the *hostgroup* format description. New host group entries can be created and existing can be modified via the **-ahgrp**, **-mhgrp**, **-dhgrp** and **-?attr** options to qconf(1).

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

A host group entry contains following parameters:

group_name

The group_name defines the host group name. Host group names have to begin with an '@' character as explained for hostgroup_name in sge_types(1).

hostlist

The name of all hosts and host groups (see host_identifier in sge_types(1)) which are member of the group. As list separators white-spaces are supported only. Default value for this parameter is NONE.

Note, if the first character of the host_identifier is an "@" sign the name is used to reference a sge_hostgroup(5) which is taken as sub group of this group.

EXAMPLE

This is a typical host group entry:

group_name @bigMachines hostlist @solaris64 @solaris32 fangorn balrog

The entry will define a new host group called **@bigMachines**. In this host group are the host **fangorn**, **balrog** and all members of the host groups **@solaris64** and **@solaris32**.

SEE ALSO

sge_types(1), qconf(1)

COPYRIGHT

sge_intel(5)

NAME

sge_intel - Management of Intel GPUs in Altair Grid Engine

DESCRIPTION

The sections below describe the setup and management of Intel GPUs in Altair Grid Engine.

OVERVIEW

Altair Grid Engine uses Resource Maps (complex with type `RSMAP`) to map physical GPUs to internal resources that can be used during the scheduling of jobs.

For an overview of Resource Maps and how to configure and request complexes see sge_resource_map(5) and sge_complex(5).

Intel XPU Manager (XPUM)

Altair Grid Engine uses Intel XPU Manager (XPUM) to get information about installed Intel GPUs.

If installed, Altair Grid Engine will automatically load the XPUM library and set the load value m_gpu (i.e. the number of installed physical GPUs) and other GPU specific load values that can be displayed via `qconf -se <hostname>`.

It is also used to check the health of the installed GPUs and to collect GPU specific usage values of jobs.

For a list of supported XPUM versions see the Release Notes.

Health Checks: XPUM performs basic health checks of GPUs. If XPUM reports a health problem for one of the installed GPUs, the load values “xpu.<gpu>.health” and “xpu.<gpu>.health_message” will be shown in `qconf -se <hostname>`. A GPU with a health value != 0 will not be used for scheduling.

Job Usage Statistics: XPUM can be used to collect GPU usage statistics of jobs. For more information see USAGE STATISTICS.

GPU Load Values

If XPUM is available, the following GPU specific load values will be reported for hosts with supported Intel GPUs. The load values are displayed in `qconf -se <hostname>`.

Value	Description
xpu.devices	Number of installed Intel GPUs
xpu.verstr	Installed driver version
xpu.<gpu>.affinity	Affinity between GPU <gpu> and the installed CPU(s) as reported by `xpumcli topology`
xpu.<gpu>.gpu_temp	GPU temperature
xpu.<gpu>.health	Health status of the GPU. Only reported when != 0
xpu.<gpu>.health_message	Health status message of the GPU
xpu.<gpu>.mem_total	Total memory of the GPU
xpu.<gpu>.mem_free	Free memory of the GPU
xpu.<gpu>.mem_used	Used memory of the GPU
xpu.<gpu>.name	Name of the GPUs
xpu.<gpu>.path	Device path of the MIG (needed for cgroups)
xpu.<gpu>.power_usage	Current power usage of the GPU
xpu.<gpu>.uuid	UUID of the GPU

SETUP

Resource Maps

For an overview of Resource Maps and how to configure complexes with type `RSMAP` see sge_resource_map(5).

The following additional parameter can be used to map GPUs to Resource Map ids:

- `xpu_id`: Id of the GPU to which this id should be mapped (according to the output of `xpumcli discovery`). The id will be used to export `ZE_AFFINITY_MASK`.

Cgroups

Access to GPUs can be managed via cgroups to ensure that jobs are only able to access devices that were assigned to them.

If the cgroups_param **devices** is set to a list of device paths (separated by `|`), the cgroups devices subsystem/controller will be used to block access to GPUs in the list and jobs will only be able to access GPUs that were assigned to them via RSMAPs (see RMSAP parameter `device`)

The following configuration will block access to all Intel GPUs:

```
devices=/dev/dri/card[0-256]|/dev/dri/renderD*
```

If a RSMAP with the device `/dev/dri/card1|/dev/dri/renderD128` is assigned to a job, the job will be able to access the GPU to which this device path belongs. The device paths for each GPU are shown in the `qconf -se <hostname>` output.

Jobs that do not request RSMAPs will not be able to access any of the installed GPUs.

For instructions on how to enable cgroups and how to set the cgroup_param **devices**, see sge_conf(5).

SCHEDULING

Requesting GPUs

For a overview of Resource Maps and how to request complexes with type `RSMAP` see sge_resource_map(5).

The following additional parameters can be requested for GPUs and will have an effect on the scheduling result:

- `affinity`: see GPU-CPU Affinity

GPU-CPU Affinity

If a job requests a GPU and **affinity**, it will automatically be bound to the CPU cores that have a good affinity to the assigned GPU (see `xpumcli topology`). This ensures that the data between the CPU and GPU is transferred in the fastest way possible.

The parameter supports two values:

- **1/true**: Affinity is requested as hard request
- **2**: Affinity is requested as a soft request

If affinity is requested as hard request, the job will not be scheduled unless there is a GPU/CPU pair with good affinity and enough free CPU cores available. If affinity is requested as a soft request, Altair Grid Engine will try to schedule the job with a GPU/CPU pair with good affinity, but schedule the job anyway without binding to any CPU cores, if not enough CPU cores are available and free. If less cores are needed the request can be combined with the **-binding** switch.

ZE_AFFINITY_MASK

Altair Grid Engine can automatically export the environment variable `ZE_AFFINITY_MASK` for jobs that request GPUs. For more information see sge_conf(5), `SET_ZE_AFFINITY_MASK`.

USAGE STATISTICS

If enabled, XPUM is used to collect GPU specific usage values that are displayed in the gpu_job_usage section of qstat -j <job_id> and qacct -j <job_id>.

GPU Job Usage

The following usage values are available. The values are displayed in qstat -j <job_id> in the format <hostname>(xpu.<gpu>.<usage_name>=<usage_value>).

Usage value	Description
memSize	Device memory size in bytes allocated by this process
sharedMemSize	The size of shared device memory mapped into this process

Which values to report is defined via the global/local configuration attribute *gpu_job_usage*. The following values can be configured, the default value is *none*.

Setting	Description
none	No gpu usage values will be reported
all	All implemented usage values will be reported, see the table above
<variable list>	Comma or space separated list of variable names, without *xpu.<gpu>* e.g. *memSize*, *sharedMemSize*, ...

GPU Job Accounting

GPU usage values can also be written to the accounting file and displayed in qacct -j <job_id>.

Which values to write to the accounting file is defined via the global/local configuration attribute *gpu_job_accounting*.

The default value for **gpu_job_accounting** is *none*. All settings that are valid for **gpu_job_usage** are also valid for **gpu_job_accounting**.

NOTE: **gpu_job_usage** and **gpu_job_accounting** can be configured individually. If **gpu_job_accounting** is set to "none", no gpu specific usage values will be written to the accounting file, regardless of the configuration of **gpu_job_usage**.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), qsub(1), complex(5), host(5), sge_job_class(5), sge_resource_map(5).

COPYRIGHT

sge_job_class

NAME

Job Class - job class entry file format

DESCRIPTION

When Altair Grid Engine jobs are submitted then various submit parameters have to be specified either as switches which are passed to command line applications or through corresponding selections in the graphical user interface. Some of those switches define the essential characteristics of the job, others describe the execution context that is required so that the job can be executed successfully. Another subset of switches needs to be specified only to give Altair Grid Engine the necessary hints on how to handle a job correctly so that it gets passed through the system quickly without interfering with other jobs.

In small and medium sized clusters with a limited number of different job types this is not problematic. The number of arguments that have to be specified can either be written into default request files, embedded into the job script, put into an option file (passed with **@** of qsub(1)) or they can directly be passed at the command line.

Within larger clusters or when many different classes of jobs should run in the cluster then the situation is more complex and it can be challenging for a user to select the right combination of switches with appropriate values. Cluster managers need to be aware of the details of the different job types that should coexist in the cluster so that they can setup suitable policies in line with the operational goals of the site. They need to instruct the users about the details of the cluster setup so that these users are able to specify the required submission requests for each job they submit.

Job classes have been introduced in Altair Grid Engine 8.1 to be able to:

- specify job templates that can be used to create new jobs.
- reduce the learning curve for users submitting jobs.
- avoid errors during the job submission or jobs which may not fit site requirements.
- ease the cluster management for system administrators.
- provide more control to the administrator for ensuring jobs are in line with the cluster set-up.
- define defaults for all jobs that are submitted into a cluster.
- improve the performance of the scheduler component and thereby the throughput in the cluster.

Imagine you have users who often make mistakes specifying memory limits for a specific application called *memeater*. You want to make it easy for them by specifying meaningful defaults but you also want to give them the freedom to modify the memory limit default

according to their needs. Then you could use the following job class configuration (only an excerpt of the full configuration is shown):

```
jcname          memeater
variant_list    default,short,long
owner           NONE
user_lists      NONE
xuser_lists     NONE
...
CMDNAME         /usr/local/bin/memeater
...
l_hard          {~}{~}h\_vmem=6GB
```

Without going into the specifics of the job class syntax, the above job class will use a default of 6 GB for the memory limit of the job. It will, however, be feasible for users to modify this limit. Here are two examples for how users would submit a job based on this job class. The first maintaining the default, the second modifying it to 8 GB (again without going into the details of the syntax being used here):

1. qsub -jc memeater
2. qsub -jc memeater -l h_vmem=8GB

Now assume a slightly modified scenario where you want to restrict a certain group of users called novice to only use the preset of 6 GB while another group of users called expert can either use the default or can modify the memory limit. The following job class example would accomplish this. And the trick is that job classes support so called variants as well as user access lists:

```
jcname          memeater
variant_list    default advanced
owner           NONE
user_lists      novice,[advanced=expert]
xuser_lists     NONE
...
CMDNAME         /usr/local/bin/memeater
...
l_hard          h_vmem=6GB,[{~}advanced={~}h_vmem=6GB]
```

With this job class configuration, the novice users would only be able to submit their job using the first command example below while expert users could use both examples:

1. qsub -jc memeater
2. qsub -jc memeater.advanced -l h_vmem=8GB

The two use cases for job classes above are only snippets for all the different scenarios to which job classes may be applied and they only provide a glimpse onto the features of job classes. The next sections describe all attributes forming a job class object, commands that

are used to define job classes as well as how these objects are used during job submission to form new jobs. A set of examples with growing functionality will illustrate further use cases. This will be followed by describing how job classes can be embedded with other parts of a Altair Grid Engine configuration to extract the maximum benefit from job classes. Finally, specific means for monitoring job class jobs will be shown.

FORMAT OF JOB CLASS ATTRIBUTES

A job class is a new object type in Altair Grid Engine 8.1. Objects of this type can be defined by managers and also by users of a Altair Grid Engine cluster to prepare templates for jobs. Those objects can later on be used to create jobs.

Like other configuration objects in Altair Grid Engine each job class is defined by a set of configuration attributes. This set of attributes can be divided into two categories. The first category contains attributes defining a job class itself and the second category all those which form the template which in turn eventually gets instanti- ated into new jobs.

Following attributes describe characteristics of a job class:

jcname

The *jcname* attribute defines a name that uniquely identifies a job class. Please note that *NO_JC* and *ANY_JC* are reserved keywords that cannot be used as names for new job classes.

There is one particular job class with the special name *template*. It acts as template for all other job classes and the configuration of this job class template can only be adjusted by users having the manager role in Altair Grid Engine. This gives manager accounts control about default settings, some of which also can be set so that they must not be changed (see below for more information on how to enforce options).

variant_list

Job classes may, for instance, represent an application type in a cluster. If the same application should be started with various different settings in one cluster or if the possible resource selection applied by Altair Grid Engine system should depend on the mode how the application should be executed then it is possible to define one job class with multiple variants. A job class variant can be seen as a copy of a job class that differs only in some aspects from the original job class.

The *variant_list* job class attribute defines the names of all existing job class variants. If the keyword *NONE* is used or when the list contains only the word *default* then the job class has only one variant. If multiple names are listed here, that are separated by commas, then the job class will have multiple variants. The *default* variant always has to exist. If the *variant_list* attribute does not contain the word *default* then it will be automatically added by the Altair Grid Engine system.

Other commands that require a reference of a job class can either use the *jcname* or *jcname.default* to refer to the default variant of a job class or they can reference a different

variant by combining the *jcname* with the name of a specific variant. Both names have to be separated by a dot (.) character.

owner

The *owner* attribute denotes the ownership of a job class. As default the user that creates a job class will be the owner. Only this user and all managers are allowed to modify or delete the job class object. Managers and owners can also add additional user names to this list to give these users modify and delete permissions. If a manager creates a job class then the *owner* will be *NONE* to express that only managers are allowed to modify or delete the corresponding job class. Even if a job class is owned only by managers it can still be used to create new jobs. The right to derive new jobs from a job class can be restricted with the *user_lists* and *xuser_lists* attributes explained below.

user_lists

The *user_lists* job class parameter contains a comma separated list of Altair Grid Engine user access list names and user names. User names have to be prefixed with a percent character (%). Each user referenced in the *user_lists* and each user in at least one of the enlisted access lists has the right to derive new jobs from this job class using the **-jc** switch of one of the submit commands. If the *user_lists* parameter is set to *NONE* (the default) any user can use the job class to create new jobs if access is not explicitly excluded via the *xuser_lists* parameter described below. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to use the job class.

xuser_lists

The *xuser_lists* job class contains a comma separated list of Altair Grid Engine user access list names and user names. User names have to be prefixed with a percent character (%). Each user referenced in the *xuser_lists* and users in at least one of the enlisted access lists are not allowed to derive new jobs from this job class. If the *xuser_lists* parameter is set to *NONE* (the default) any user has access. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to use the job class.

FORMAT OF JOB CLASS ATTRIBUTES THAT FORM A JOB TEMPLATE

Additionally to the attributes mentioned previously each job class has a set of attributes that form a job template. In most cases the names of those additional attributes correspond to the names of command line switches of the qsub(1) command. The value for all these additional attributes might either be the keyword *UNSPECIFIED* or it might be the same value that would be passed with the corresponding qsub(1) command line switch.

All these additional job template attributes will be evaluated to form a virtual command line when a job class is used to instantiate a new job. All attributes for which the corresponding value contains the *UNSPECIFIED* keyword will be ignored whereas all others define the submit arguments for the new job that will be created.

All template attributes can be divided in two groups. There are template attributes that accept simple attribute values (like a character sequence, a number or the value yes or no) and there are template attributes that allow to specify a list of values or a list of key/value pairs (like the list of resource requests a job has or the list of queues where a job might get executed).

The sections below explain all available template attributes. The asterisk character (*) tags all attributes that are list based. Within the description the default for each attribute is documented that will be used when the keyword *UNSPECIFIED* is used in the job class definition.

a

Specifies the time and date when a job is eligible for execution. If unspecified the job will be immediately eligible for execution. Format of the character sequence is the same as for the argument that might be passed with qsub(1) -a.

A

Account string. The string *sge* will be used when there is no account string specified or when it is later on removed from a job template or job specification.

ac (*)

List parameter defining the name/value pairs that are part of the job context. Default is an empty list.

ar

Advance reservation identifier used when jobs should be part of an advance reservation. As default no job will be part of an advance reservation.

b

yes or *no* to express if the command should be treated as binary or not. The default for this parameter is *no*, i.e. the job is treated as a script.

binding

Specifies all core binding specific settings that should be applied to a job during execution. Binding is disabled as default.

CMDARG (*)

Defines a list of command line arguments that will be passed to *CMDNAME* when the job is executed. As default this list is empty.

CMDNAME

Specified either the job script or the command name when binary submission is enabled (*b yes*). Please note that script embedded flags within specified job scripts will be ignored.

c_interval

Defines the time interval when a checkpoint-able job should be checkpointed. The default value is *0*.

c_occasion

Letter combination that defines the state transitions when a job should be triggered to write a checkpoint. Default is *n* which will disable checkpointing.

ckpt

Checkpoint environment name which specifies how to checkpoint the job. No checkpoint object will be referenced as default.

cwd

Specifies the working directory for the job. Path aliasing will not be used when this value is specified in a job class. In case of absence the home directory of the submitting user will be used as directory where the job is executed.

dl

Specifies the deadline initiation time for a job (see the chapter about deadline urgency in the administrators guide for more information). As default jobs have do defined deadline.

e (*)

List parameter that defines the path for the error file for specific execution hosts. As default the file will be stored in the home directory of the submitting user and the filename will be the combination of the job name and the job id.

h

yes or *no* to indicate if a job should be initially in hold state. The default is *no*.

hold_jid (*)

List parameter to create initial job dependencies between new jobs and already existing ones. The default is an empty list.

hold_jid_ad (*)

List parameter to create initial array job dependencies between new array jobs and already existing ones. The default is an empty list.

i (*)

List parameter that defines the path for the input file for specific execution hosts

j

yes or *no* to show if error and output stream of the job should be joined into one file. Default is *no*.

js

Defines the job share of a job relative to other jobs. The default is *0*.

l_hard (*)

List parameter that defines hard resource requirements of a job in the form of name/value pairs. The default is an empty list.

l_soft (*)

List parameter defining soft requests of a job. The default is an empty list.

mbind

Specifies memory binding specific settings that should be applied to a job during execution. Memory binding is disabled as default.

m

Character sequence that defines the circumstances when mail that is related to the job should be send. The default is *n* which means no mails should be send.

M (*)

List parameter defining the mail addresses that will be used to send job related mail. The default is an empty list.

masterl (*)

List parameter that defines hard resource requirements for the master task of a parallel in the form of name/value pairs. The default is an empty list.

masterq (*)

List parameter that defines the queues that might be used as master queues for parallel jobs. The default is an empty list.

N

Default name for jobs. For jobs specifying a job script which are submitted with qsub or the graphical user interface the default value will be the name of the job script. When the script is read from the stdin stream of the submit application then it will be *STDIN*. qsh(1) and qlogin(1) jobs will set the job name to *INTERACTIVE*. qrsh(1) jobs will use the first characters of the command line up to the first occurrence of a semicolon or space character.

notify

yes or *no*: to define if warning signals will be send to a jobs if it exceeds any limit. The default is *no*.

now

yes or *no* to specify if created jobs should be immediate jobs. The default is *no*.

o (*)

List parameter that defines the path for the output file for specific execution hosts.

P

Specifies the project to which this job is assigned.

p

Priority value that defines the priority of jobs relative to other jobs. The default priority is *0*.

pe_name

Specifies the name of the parallel environment that will be used for parallel jobs. As default there is no name specified and as a result the job is no parallel job.

pe_range

Range list specification that defines the amount of slots that are required to execute parallel jobs. This parameter must be specified when also the pe_name parameter is specified.

q_hard (*)

List of queues that can be used to execute the job. The default is an empty list.

q_soft (*)

List of queues that are preferred to be used when the job should be executed. The default is an empty list.

R

yes or *no* to indicate if a reservation for this job should be done. The default is *no*.

r

yes or *no* to identify if the job will be rerun-able. The default is *no*.

rdi

yes if request dispatch information should be generated by the scheduler component for a job. The default is *no*.

rou (*)

List of online usage variables that shall get written to the reporting file and the reporting database. The default is an empty list.

S (*)

List parameter that defines the path of the shell for specific execution hosts. The default is an empty list.

shell

yes or *no* to specify if a shell should be executed for binary jobs or if the binary job should be directly started. The default is *yes*

t

Defines the task ID range for array jobs. Jobs are no array jobs as default.

umask

Defines the umask that influences the file permissions of output- and error-files created for a job. If unspecified the default is 0022.

V

TRUE or *FALSE*. *TRUE* causes that all environment variables active during the submission of a job will be exported into the environment of the job. With *NO* (or *UNSPECIFIED*) those variables will not be exported. In order to disallow the use of the **-V** switch in combination with job class submission a job class has to specify the keyword *UNSPECIFIED* without an access specifier.

v (*)

List of environment variable names and values that will be exported into the environment of the job. If also *V yes* is specified then the variable values that are active during the submission might be overwritten.

xd (*)

List of *-xd* switches for specifying arbitrary docker run options to be used in the creation of the container for Docker jobs. Docker run means the run option of the docker command that is part of the Docker Engine.

ACCESS SPECIFIERS

Access specifiers are character sequences that can be added to certain places in job class specifications to allow/disallow operations that can be applied to jobs that are derived from that job class. They allow you to express, for instance, that job options defined in the jobs class can be modified, deleted or augmented when submitting a job derived from a job class. This means the job class owner can control how the job class can be used by regular users being allowed to derive jobs from this job class. This makes using job classes simple for the end user (because of a restricted set of modifications). It also avoids errors as well as the need to utilize Job Submission Verifiers for checking on mandatory options.

Per default, if no access specifiers are used, all values within job classes are fixed. This means that jobs that are derived from a job class cannot be changed. Any attempt to adjust a job during the submission or any try to change a job after it has been submitted (e.g. with qalter) will be rejected. Also managers are not allowed to change the specification of attributes defined in a job class when submitting a job derived from the job class.

To soften this restriction, job class owners and users having the manager role in a job class can add access specifiers to the specification of a job class to allow deviation at certain places. Access specifiers might appear before each value of a job template attribute and before each entry in a list of key or key/value pairs. The preceding access specifier defines which operations are allowed with the value that follows.

The full syntax for a job class template attribute is defined as <jc_templ_attr>:

```
<jc_templ_attr>
        := <templ_attr_and_value> | <list_templ_attr_and_value> |
           <templ_attr_and_value_for_task_range> |
           <list_templ_attr_and_value_for_task_range> ;

<templ_attr_and_value>
        := <attr_name> ' ' <templ_attr_value> ;

<templ_attr_value>
        := [ <attr_access_specifier> ] ( <attr_value> |
                                         'UNSPECIFIED' ) ;

<list_templ_attr_and_value>
        := <list_attr_name> ' ' <list_attr_value>) ;

<list_attr_value>
        := [ <attr_access_specifier> ]
           ( ( [ <access_specifier> ] <list_entry>
               { ',' [ <access_specifier> ] <list_entry> } )
             | 'UNSPECIFIED' ) ;

<templ_attr_and_value_for_task_range>
        := <attr_name> ' ' <templ_attr_value>
           { ';' <task_id_range> '|' <templ_attr_value> } ;
```

```
<list_templ_attr_and_value_for_task_range>
        := <list_attr_name> ' ' <list_attr_value>
           { ';' <task_id_range> '|' <list_attr_value> } ;

<attr_access_specifier>
        := <access_specifier> ;
```

<templ_attr_and_value_for_task_range> and <list_templ_attr_and_value_for_task_range> that allow to specify task specific resource requests for parallel jobs and that therefore contain task ID ranges (<task_id_range>) are explained further below.

Please note the distinction between <attr_access_specifier> and <access_specifier>. <attr_access_specifier> is also an <access_specifier> but it is the first one that appears in the definition of list based job template attributes and it is the reason why two access specifiers might appear one after another. The first access specifier regu- lates access to the list itself whereas the following ones define access rules for the first entry in the list. These access specifiers (<access_specifier>) are available:

(no access specifier)

The absence of an access specifier indicates that the corresponding template attribute (or sublist entry) is fixed. Any attempt to modify or delete a specified value or any attempt to add a value where the keyword *UNSPECIFIED* was used will be rejected. It is also not allowed to add additional entries to lists of list based attributes if a list is fixed.

{-}

Values that are tagged with the {-} access specifier are removable. If this access specifier is used within list based attributes then removal is only allowed if the list itself is also modifiable. If all list entries of a list are removable then also the list itself must be removable so that the oper- ation will be successful. Values that are tagged with the {-} access specifier are removable.

{~}

Values that are prefixed with the {~} access specifier can be changed. If this access specifier is used within list based attributes then the list itself must also be modifiable.

{-~} or {~-}

The combination of the {-} and {~} access specifiers indicates that the value it precedes is modifiable and removable.

{+}UNSPECIFIED or {+...}<list_attr_value>

The {+} access specifier can only appear in combination with the keyword *UNSPECIFIED* or before list attribute values but not within access specifiers preceding list entries. If it appears before list attribute values it can also be combined with the {~} and {-} access specifiers. This access specifier indicates that something can be added to the specification of a job after it has been submitted. For list based attributes it allows that new list entries can be added to the list.

JOB CLASS VARIANTS

Job classes represent an application type in a cluster. If the same application should be started with various different settings or if the possible resource selection applied by the Altair Grid Engine system should depend on the mode how the application should be executed then it is possible to define one job class with multiple variants. So think of it as a way to use the same template for very similar types of jobs, yet with small variations.

The *variant_list* job class attribute defines the names of all existing job class variants. If the keyword *NONE* is used or when the list contains only the word *default* then the job class has only one variant. If multiple names are listed here, separated by commas, then the job class will have multiple variants. The default variant always has to exist. If the *variant_list* attribute does not contain the word default then it will be automatically added by the Altair Grid Engine system upon creating the job class.

Attribute settings for the additional job class variants are specified similar to the attribute settings of queue instances or queue domains of cluster queues. The setting for a variant attribute has to be preceded by the variant name followed by an equal character (=) and enclosed in brackets ("[" and "]"). The position where access specifiers have to appear is slightly different in this case. The example section of this man page will show this.

TASK SPECIFIC REQUESTS FOR PARALLEL JOBS

Job classes that act as a template for parallel jobs (where *pe_name* and *pe_range* is specified) allow parallel task specific deviations for the attributes *l_hard*, *l_soft*, *q_hard*, *q_soft* and *par*. For those attributes the attribute syntax is enhanced in order to be able to specify task specific deviations for an individual task or single task range.

In such a case the attribute setting can be prefixed by a task range followed by a pipe-character ('|') immediately before access specifier and value are specified. Multiple task specific settings have to be separated by a semicolon-character (';'). Please note that in difference to the command line, comma separated task IDs or task ID range lists are not allowed within job class definitions. Corresponding resource requests for those tasks or ranges of tasks have to be specified separately.

As soon as a task specific setting is specified in a job class also a non task specific setting has to be defined that will be used as default for those tasks only which have no specific values defined. This default setting can be the *UNSPECIFIED* keyword.

Omitting a range number ('1-') or specifying the step size for an open ended range (e.g. '1-:2') is allowed as long as the settings are unique for each possible task number. Only one specific setting for a task is allowed otherwise a job class definition will be rejected.

Following job class with two variants has task specific settings for tasks. Assume that A, B, C, Y and Z are simple boolean complexes and that Q is a queue name.

```
jcname       jc
variant_list default,large
pe_name      pe1
pe_range     8
 ...
 q_hard       {+}UNSPECIFIED;
              0|{+~}{~}Q
l_hard       {~+}{~}Z=true,[{~+}large={~}Y=true];
             0|{~+}{~}A=true;
             1-2|{~+}{~}B=true;
             3-4|{+}UNSPECIFIED,[{~+}large={~}B=true]
l_soft       {+}UNSPECIFIED;
              1-2|{~+}{~}C=true
...
```

Following two jobs have the same resource requests.

```
qsub -jc jc.default
```

```
qsub -pe pe1 8 ...
     -l Z
     -petask 0 -q Q -l A
     -petask 1-2 -l B -soft -l C
     ...
```

Also the next two jobs would have the same resource requests

```
qsub -jc jc.large
```

```
qsub -pe pe1 8 ...
     -l Y
     -petask 0 -q Q -l A
     -petask 1-4 -l B
     ...
```

The structure of job classes and the fact that multiple variants can be specified in one job class definition might make it necessary to split simple resource definitions used in a submission command line in multiple parts within a job class definition. Complexity of the job class definition will increase due to that. To reduce that complexity it will help to avoid the use of task ranges in job class definitions by specifying the resource requests for single tasks individually or by using multiple job classes to define different job variants instead of using job class variants within one job class.

CLUSTER WIDE REQUESTS WITH THE TEMPLATE JOB CLASS

After a default installation of Altair Grid Engine 8.1 there exists one job class with the name *template*. This job class has a special meaning and it cannot be used to create new jobs. Its configuration can only be adjusted by users having the manager role. This jobs class acts as parent job class for all other job classes that are created in the system.

The values of job template attributes in this template job class and the corresponding access specifiers restrict the allowed settings of all corresponding job template attributes of other job classes. As default the {+}*UNSPECIFIED* add access specifier and keyword is used in the *template* job class in combination with all job template attributes. Due to that any setting is allowed to other job class attributes after Altair Grid Engine 8.1 has been installed.

This parent-child relationship is especially useful when all jobs that are submitted into a cluster are derived from job classes. Managers might then change the settings within the *template*. All other existing job classes that violate the settings will then switch into the *configuration conflict* state. The owners of those job classes have to adjust the settings before new jobs can be derived from them. All those users that intend to create a new job class that violates the settings of the template job class will receive an error.

You will also want to use the *template* job class to enforce restrictions on the access specifiers which can be used in job classes. Since any job class, whether created by a manager account or by regular users, is derived from the template job class those derived job classes are bound to stay within the limits defined by the *template* job class. So parameters which have been defined as fixed in the *template* job class, for instance, cannot be modified in any job class created by a manager or user. Likewise, parameters which have a preset value but are configured to allow deletion only cannot be modified in derived job classes. The following table shows the allowed transitions:

AS in Template	Allowed AS in Child JC
<attr_value>	<attr_value>
UNSPECIFIED	UNSPECIFIED
{~}<attr_value>	{~}<attr_value>
{~}<attr_value>	<attr_value>
{-}<attr_value>	{-}<attr_value>
{-}<attr_value>	{~}<attr_value>
{-}<attr_value>	UNSPECIFIED
{-}<attr_value>	<attr_value>
{-~}<attr_value>	{-~}<attr_value>
{-~}<attr_value>	{-}<attr_value>
{-~}<attr_value>	{~}<attr_value>
{-~}<attr_value>	UNSPECIFIED
{-~}<attr_value>	<attr_value>
{+}...	{+}...
{+}...	{-~}<attr_value>
{+}...	{-}<attr_value>

AS in Template	Allowed AS in Child JC
{+}...	{~}<attr_value>
{+}...	UNSPECIFIED
{+}...	<attr_value>

RELATIONSHIP TO OTHER OBJECTS

To fully integrate job classes into the already existing Altair Grid Engine system the possibility is provided to create new relations between current object types (like queues, resource quotas, JSV) and job classes.

RESOURCES AVAILABLE FOR JOB CLASSES

The profile of a job is defined by the resource requirements and other job attributes. Queues and host objects define possible execution environments where jobs can be executed. When a job is eligible for execution then the scheduler component of the Altair Grid Engine system tries to find the execution environment that fits best according to all job specific attributes and the configured policies so that this job can be executed.

This decision making process can be difficult and time consuming especially when certain jobs having special resource requirements should only be allowed to run in a subset of the available execution environments. The use of job classes might help here because job classes will give the scheduler additional information on which execution environments will or will not fit for a job. The need to evaluate all the details about available resources of an execution environment and about the job's requirements will be reduced or can be completely eliminated during the decision making process.

This is achieved by an additional paramter in the queue configuration which provides a direct association between queues and one or multiple job classes. This paramter is called *jc_list* and might be set to the value *NONE* or a list of job classes or job class variant names. If a list of names is specified then the special keyword *ANY_JC* and/or *NO_JC* might be used within the list to filter all those jobs that are in principle allowed to run in these queues. The following combinations are useful:

Value	Description
NONE	No job my enter the queue.
ANY_JC	Jobs may enter the queue that were derived from a job class.
NO_JC	Only jobs may enter the queue that were not derived from a job class.
ANY_JC, NO_JC	Any job, independent if it was derived from a job class or not, may be executed in the queue. This is the default for any

Value	Description
	queue that is created in a cluster.
<list of JC names>	Only those jobs may get scheduled in the queue if they were derived from one of the enlisted job classes.
NO_JC, <list of JC names>	Only those jobs that were not derived from a job class or those that were derived from one of the enlisted job classes can be executed here.

This relationship helps the scheduler during the decision making to eliminate queues early without the need to further look at all the details like resource requirements. Managers of Altair Grid Engine Clusters may want to take care that there is at least one queue in the cluster available that use the *ANY_JC* keyword. Otherwise jobs of users who have defined their own job class will not get cluster resources. Also at least one queue using the *NO_JC* keyword may need to be available. Otherwise conventionally submitted jobs will not get scheduled.

The *jc_list* defined in queues influences also which resources are considered for the resource reservation done in combination with advance reservations.

RESOURCES CONSIDERED FOR ADVANCE RESERVATIONS

Advance reservations allow to reserve resources for jobs that should later on be executed within that advance reservation. The **-jc** switch available for the qrsub(1) command gives the creator of an AR the chance to influence which queue instances might be reserved as part of the reservation process.

If no special job class is selected or when **-jc** is used in combination with the *IGNORE_JC* keyword then the queue selection algorithm will select only those queues that allow the execution of JC jobs *AND* non-JC jobs.

Alternatively the keywords *NO_JC* or *ANY_JC* can be specified during the AR submission to select only queues that do not allow or allow JC jobs. As consequence only corresponding jobs can be executed within that AR later on.

It is also possible to select a job class variant or a pattern that matches multiple job class variants so that only those queue instances will be selected during the reservation process. Also only corresponding jobs are allowed to be executed.

DEFINING JOB CLASS LIMITS

Resource quota sets can be defined to influence the resource selection in the scheduler. The *jcs* filter within a resource quota rule may contain a comma separated list of job class names. This parameter filters for jobs requesting a job class in the list. Any job class not in the list will not be considered for the resource quota rule. If no *jcs* filter is used, all job

classes and jobs with no job class specification match the rule. To exclude a job class from the rule, the name can be prefixed with the exclamation mark (!). !* means only jobs with no job class specification.

JSV AND JOB CLASS INTERACTION

During the submission of a job multiple Job Submission Verifiers can be involved that verify and possibly correct or reject a job. With conventional job submission (without job classes) each JSV will see the job specification of a job that was specified at the command line via switches and passed parameters or it will see the job parameters that were chosen within the dialogs of the GUI.

When Jobs are derived from a job class then the process of evaluation via JSV scripts is the same but the job parameters that are visible in client JSVs are different. A client JSV will only see the requested job class via a parameter named *jc* and it will see all those parameters that were specified at the command line. All parameters that are defined in the job class itself cannot be seen.

Job classes will be resolved within the sge_qmaster(8) process as soon as a request is received that tries to submit a job that should be derived from a job class. The following steps are taken (simplified process):

1. Create a new job structure
2. Fill job structure with defaults values
3. Fill job structure with values defined in the job class. (This might overwrite default values)
4. Fill job structure with values defined at the command line. (This might overwrite default values and values that were defined in the job class)
5. Trigger server JSV to verify and possibly adjust the job. (This might overwrite default values, JC values and values specified at the command line)
6. Check if the job structure violates access specifiers.

If the server JSV changes the *jc* parameter of the job in step 5 then the submission process restarts from step 1 using the new job class for step 3.

Please note that the violation of the access specifiers is checked in the last step. As result a server JSV is also not allowed to apply modifications to the job that would violate any access specifiers defined in the job class specification.

ENFORCE THE USE OF JOB CLASSES

The global configuration object of Altair Grid Engine allows to define two attributes (*default_jc*, *enforce_jc*) to enforce the use of job classes. Find more in formation in sge_conf(5).

STATES OF JOB CLASSES

Job Classes have a combined state that is the result of following sub states: *enabled*/*disabled*, *no_conflict*/*configuration_conflict*.

The *enabled*/*disabled* state is a manual state. A state change from *enabled* to *disabled* can be triggered with the qmod(1) **-djc** command. The command qmod(1) **-ejc** command can be used to trigger a state change from *disabled* to *enabled*. Job Classes in the *disabled* state cannot be used to create new jobs.

The *no_conflict*/*configuration_conflict* state is an automatic state that cannot be changed manually. Job classes that do not violate the configuration of the *template* job class are in the *no_conflict* state. A job class in this state can be used to create new jobs (if it is also in *enabled* state). If the *template* job class or a derived job class is changed so that either a configuration setting or one of the access specifiers of the *template* job class is violated then the derived job class will automatically switch from the *no_conflict* into the *configuration_conflict* state. This state will also be left automatically when the violation is eliminated.

USING JOB CLASSES TO SUBMIT NEW JOBS

Job Classes that are in the enabled and no conflict state can be used to create new jobs. To do this a user has to pass the *-jc* switch in combination with the name of a job class to a submit command like qsub(1). If the user has access to this job class then a new job will be created and all job template attributes that are defined in the job class will be used to initialize the corresponding parameters in the submitted job.

Depending on the access specifiers that are used in the job class it might be allowed to adjust certain parameters during the submission of the job. In this case additional switches and parameters might be passed to the submit command. All these additionally passed parameters will be used to adjust job parameters that where derived from the job class.

Additional to the typical switches that are used to define job parameters there is a set of switches available that allow to remove parameters or to adjust parts of list based parameters in a job specification. The same set of switches can also be used with the modification command qalter(1) to adjust job parameters after a job has already been created.

qsub/qalter -clearp *attr_name*

The **-clearp** switch allows to remove a job parameter from the specification of a job as if it was never specified. What this means depends on the job parameter that is specified by *attr_name*. For all those attributes that would normally have a default value this default value will be set for all others the corresponding attribute will be empty. Parameter names that can be specified for *attr_name* are all the ones that are specified in the table above showing job template attribute names.

qsub/qalter -clears *list_attr_name key*

This switch allows to remove a list entry in a list based attribute of a job specification. *list_attr_name* might be any name of a job template attribute that is tagged with the asterisk (*) in the the table above. *key* has to be the name of the key of the sublist entry for key/value pairs or the value itself that should be removed when the list contains only values.

qsub/qalter -adds *list_attr_name key value*

-adds adds a new entry to a list based parameter.

qsub/qalter -mods *list_attr_name key value*

The **-mods** switch allows to modify the value of a key/value pair within a list based job parameter.

EXAMPLE

Assume that the following job class is defined in you cluster:

```
jcname          sleeper
variant_list    default,short,long
owner           NONE
user_lists      NONE
xuser_lists     NONE
A               UNSPECIFIED
a               UNSPECIFIED
ar              UNSPECIFIED
b               yes
binding         UNSPECIFIED
c_interval      UNSPECIFIED
c_occasion      UNSPECIFIED
CMDNAME         /bin/sleep
CMDARG          60,[short=5],[long=3600]
ckpt            UNSPECIFIED
ac              UNSPECIFIED
cwd             UNSPECIFIED
display         UNSPECIFIED
dl              UNSPECIFIED
e               UNSPECIFIED
h               UNSPECIFIED
hold_jid        UNSPECIFIED
i               UNSPECIFIED
j               UNSPECIFIED
js              UNSPECIFIED
l_hard          {~+}{~}a=true,b=true,{-}c=true
l_soft          {+}UNSPECIFIED,[{~+}long={~}d=true]
m               UNSPECIFIED
M               UNSPECIFIED
masterq         UNSPECIFIED
masterl         UNSPECIFIED
mbind           UNSPECIFIED
N               {~-}Sleeper,[{~-}short=ShortSleeper],[{~-}long=LongSleeper]
```

```
notify          UNSPECIFIED
now             UNSPECIFIED
o               UNSPECIFIED
P               UNSPECIFIED
p               UNSPECIFIED
pe_name         UNSPECIFIED
q_hard          UNSPECIFIED
q_soft          UNSPECIFIED
R               UNSPECIFIED
r               UNSPECIFIED
S               /bin/sh
shell           UNSPECIFIED
V               UNSPECIFIED
v               UNSPECIFIED
```

Now it is possible to submit jobs and to adjust the parameters of those jobs during the submission to fit specific needs:

1. qsub -jc sleeper -N MySleeper
2. qsub -jc sleeper.short -clearp N
3. qsub -jc sleeper.short -clears l_hard c -adds l_hard h_vmem 5G
4. qsub -jc sleeper.long -soft -l res_x=3

The first job that is submitted (1) will be derived from the *sleeper.default* job class variant but this job will get the name MySleeper.

Job (2) uses the *sleeper.short* job class but the job name is adjusted. The **-clearp** switch will remove the job name that is specified in the job class. Instead it will get the default job name that would have been assigned without specifying the name in any explicit way. This will be derived from the last part of the script command that will be executed. This script is */bin/sleep*. So the job name of the new job will be *sleep*.

When job (3) is created the list of hard resource requirements is adjusted. The resource request _c _is removed and the *h_vmem=5G* resource request is added.

During the submission of job (4) The list of soft resource request is completely redefined. The use of the **-l** will completely replace already defined soft resource requests if any have been defined. Please note that it is not allowed to trigger operations that would violate any access specifiers. In consequence, the following commands would be rejected:

5. qsub -jc sleeper -hard -l res_x 3
6. qsub -jc sleeper /bin/my_sleeper 61

Job (5) would remove the *a* and *b* resource requests and job (6) will be rejected because neither *CMDNAME* nor *CMDARGS* are modifiable.

SEE ALSO

sge_types(1), qconf(1) qsub(1) qmod(1) qalter(1)

COPYRIGHT

sge_jsv

NAME

JSV - Altair Grid Engine Job Submission Verifier

DESCRIPTION

JSV is an abbreviation for Job Submission Verifier. A JSV is a script or binary that can be used to verify, modify or reject a job during the time of job submission.

JSVs will be triggered by submit clients like qsub, qrsh, qsh and qmon on submit hosts (Client JSV) or they verify incoming jobs on the master host (Server JSV) or both.

CONFIGURATION

JSVs can be configured on various locations. Either a **jsv_url** can be provided with the **-jsv** submit parameter during job submission, a corresponding switch can be added to one of the **sge_request** files or a **jsv_url** can be configured in the global cluster configuration of the Altair Grid Engine installation.

All defined JSV instances will be executed in following order:

```
1. qsub -jsv ...
2. $cwd/.sge_request
3. $HOME/.sge_request
4. $SGE_ROOT/$SGE_CELL/common/sge_request
5. Global configuration
```

The Client JSVs (1-3) can be defined by Altair Grid Engine end users whereas the client JSV defined in the global **sge_request** file (4) and the server JSV (5) can only be defined by the Altair Grid Engine administrators.

Due to the fact that (4) and (5) are defined and configured by Altair Grid Engine administrators and because they are executed as last JSV instances in the sequence of JSV scripts, an administrator has an additional way to define certain policies for a cluster.

As soon as one JSV instance rejects a job the whole process of verification is stopped and the end user will get a corresponding error message that the submission of the job has failed.

If a JSV accepts a job or accepts a job after it applied several modifications then the following JSV instance will get the job parameters including all modifications as input for the verification process. This is done as long as either the job is accepted or rejected.

Find more information how to use Client JSVs in *qsub*(1) and for Server JSVs in *sge_conf*(5)

LIFETIME

A Client or Server JSV is started as own UNIX process. This process communicates either with a Altair Grid Engine client process or the master daemon by exchanging commands, job parameters and other data via stdin/stdout channels.

Client JSV instances are started by client applications before a job is sent to qmaster. This instance does the job verification for the job to be submitted. After that verification the JSV process is stopped.

Server JSV instances are started for each worker thread part of the qmaster process (for version 6.2 of Altair Grid Engine this means that two processes are started). Each of those processes have to verify job parameters for multiple jobs as long as the master is running, the underlying JSV configuration is not changed and no error occurs.

TIMEOUT

The timeout is a modifiable value that will measure the response time of either the client or server JSV. In the event that the response time of the JSV is longer than timeout value specified, this will result in the JSV being re-started. The server JSV timeout value is specified through the qmaster parameter **jsv_timeout**. The client JSV timeout value is set through the environment variable **SGE_JSV_TIMEOUT**. The default value is **10** seconds, and this value must be greater than **0**. If the timeout has been reach, the JSV will only try to re-start once, if the timeout is reached again an error will occur.

THRESHOLD

The threshold value is defined as a qmaster parameter **jsv_threshold**. This value measures the time for a server job verification. If this time exceeds the defined threshold then additional logging will appear in the master message file at the INFO level. This value is specified in milliseconds and has a default value of **5000**. If a value of **0** is defined then this means all jobs will be logged in the message file.

PROTOCOL

After a JSV script or binary is started it will get commands through its stdin stream and it has to respond with certain commands on the stdout stream. Data which is send via the stderr stream of a JSV instance is ignored. Each command which is send to/by a JSV script has to be terminated by a new line character ('\n') whereas new line characters are not allowed in the whole command string itself.

In general commands which are exchanged between a JSV and client/qmaster have following format. Commands and arguments are case sensitive. Find the EBNF command description below.

command := command_name ' ' { argument ' ' } ;

A command starts with a **command_name** followed by a space character and a space separated list of arguments.

JSV side of the protocol

Following commands have to be implemented by an JSV script so that it conforms to version 1.0 of the JSV protocol which was first implemented in Altair Grid Engine 6.2u2:

begin_command

begin_command := 'BEGIN' ;

After a JSV instance has received all *env_commands* and *param_commands* of a job which should be verified, the client/qmaster will trigger the verification process by sending one *begin_command*. After that, it will wait for *param_commands* and *env_commands* which are sent back from the JSV instance to modify the job specification. As part of the verification process a JSV script or binary has to use the *result_command* to indicate that the verification process is finished for a job.

env_command

env_command := 'ENV' ' ' modifier ' ' name ' ' value ;

modifier := 'ADD' | 'MOD' | 'DEL' ;

The *env_command* is an optional command which has only to be implemented by a JSV instance if the *send_data_command* is sent by this JSV before a the *started_command* was sent. Only in that case the client or master will use one or multiple *env_commands* to pass the environment variables (name and value) to the JSV instance which would be exported to the job environment when the job would be started. Client and qmaster will only sent *env_commands* with the modifier 'ADD'.

JSV instances modify the set of environment variables by sending back *env_commands* and by using the modifiers ADD, MOD and DEL.

param_command

param_command := 'PARAM' ' ' param_parameter ' ' value ;

param_parameter := submit_parameter | pseudo_parameter ;

The *param_command* has two additional arguments which are separated by space characters. The first argument is either a *submit_parameter* as it is specified in *qsub*(1) or it is a *pseudo_parameters* as documented below. The second parameter is the value of the corresponding param_parameter.

Multiple param_commands will be sent to a JSV instance after the JSV has sent a *started_command*. The sum of all *param_commands* which is sent represents a job specification of that job which should be verified.

submit_parameters are for example *b* (similar to the *qsub*(1) **-b** switch) or *masterq* (similar to *qsub*(1) **-masterq** switch). Find a complete list of *submit_parameters* in the *qsub*(1) man page. Please note that not in all cases the *param_parameter* name and the corresponding value format is equivalent with the qsub switch name and its argument format. E.g. the *qsub*(1) **-pe** parameters will by available as a set of parameters with the names *pe_name*, *pe_min*, *pe_max* or the switch combination **-soft -l** will be passed to JSV scripts as *l_soft* parameter. For details concerning this differences consult also the *qsub*(1) man page.

start_command

start_command := 'START' ;

The *start_command* has no additional arguments. This command indicates that a new job verification should be started. It is the first command which will be sent to JSV script after it has been started and it will initiate each new job verification. A JSV instance might trash cached values which are still stored due to a previous job verification. The application which send the *start_command* will wait for a *started_command* before it continues.

quit_command

quit_command := 'QUIT' ;

The *quit_command* has no additional arguments. If this command is sent to a JSV instance then it should terminate itself immediately.

Client/qmaster side of the protocol

A JSV script or binary can send a set of commands to a client/qmaster process to indicate its state in the communication process, to change the job specification of a job which should be verified and to report messages or errors. Below you can find the commands which are understood by the client/qmaster which will implement version 1.0 of the communication protocol which was first implemented in Altair Grid Engine 6.2u2:

error_command

error_command := 'ERROR' message ;

Any time a JSV script encounters an error it might report it to the client/qmaster. If the error happens during a job verification the job which is currently verified will be rejected. The JSV binary or script will also be restarted before it gets a new verification task.

log_command

log_command := 'LOG' log_level ;

log_level := 'INFO' | 'WARNING' | 'ERROR' ;

log_commands can be used whenever the client or qmaster expects input from a JSV instance. This command can be used in client JSVs to send information to the user submitting the job. In client JSVs all messages, independent of the *log_level* will be printed to the stdout stream of the used submit client. If a server JSV receives a *log_command* it will add the received message to the message file respecting the specified *log_level*. Please note that message might contain spaces but no new line characters.

param_command

(Find definition above.)

By sending *param_commands* a JSV script can change the job specification of the job which should be verified. If a JSV instance later on sends a *result_command* which indicates that a JSV instance should be accepted with correction then the values provided with these param_commands will be used to modify the job before it is accepted by the Altair Grid Engine system.

result_command

result_command := 'RESULT' result_type [message] ;

result_type := 'ACCEPT' | 'CORRECT' | 'REJECT' | 'REJECT_WAIT' ;

After the verification of a job is done a JSV script or binary has to send a result_command which indicates what should happen with the job. If the **result_type** is *ACCEPTED* the job will be accepted as it was initially submitted by the end user. All *param_commands* and *env_commands* which might have been sent before the *result_command* are ignored in this case. The *result_type CORRECT* indicates that the job should be accepted after all modifications sent via *param_commands* and *env_commands* are applied to the job. *REJECT* and *REJECT_WAIT* cause the client or qmaster instance to reject the job.

send_data_command

send_data_command := 'SEND' data_name ;

data_name := 'ENV';

If a client/qmaster receives a *send_env_command* from a JSV instance before a *started_command* is sent, then it will not only pass job parameters with *param_commands* but also env_commands which provide the JSV with the information which environment variables would be exported to the job environment if the job is accepted and started later on.

The job environment is not passed to JSV instances as default because the job environment of the end user might contain data which might be interpreted wrong in the JSV context and might therefore cause errors or security issues.

started_command

started_command := 'STARTED' ;

By sending the *started_command* a JSV instance indicates that it is ready to receive *param_commands* and *env_commands* for a new job verification. It will only receive env_commands if it sends a *send_data_command* before the *started_command*.

JOB PARAMETERS

Job parameters are parameters that describe the job to be submitted. In most cases the parameter names and values within JSV are equivalent with the name of the corresponding *qsub*(1) submit switch (*b* similar to the *qsub*(1) **-b** switch, ...).

Please note that this is not always true. The **-pe** switch is an example where this is not the case. Here multiple parameters will be exported to the JSV script (**pe_name**, **pe_n**, **pe_min_<ID>**, **pe_max_<ID>**, ...) that allow to distinguish between the specified parallel environment name, minimum and maximum slot range values of multiple ranges that might be specified with *qsub*(1) **-pe**.

There are also cases where the specification of multiple switches is summarized and less job parameters appear within JSV. *qsub*(1) **-soft** and the use of (multiple) **-l** switches will appear in JSV as **l_soft** parameter containing the full list of soft resource requests.

Also for PE task specific requests where the **-petask** switch is combined with **-soft** and **-hard** in combination with **-l**, **-q** and **-par** the commandline arguments are summarized differently. **petask_max** provides the information how many task specific requests where specified. **petask_<ID>_ranges** shows the amount of task-id ranges within one specific task range. The min, max and step-size value of a range can then be requested with the parameters **petask_<ID>_<RANGE>_min**, **petask_<ID>_<RANGE>_max** and **petask_<ID>_<RANGE>_step** where the corresponding **<ID>** and **<RANGE>** within parameter names denote the corresponding positions. Numbering starts with 0 and **<ID>** and **<RANGE>** may not exceed the corresponding maximum number minus one. Specific hard/soft requests and adapted allocation rules for specific tasks can be retrieved the same way by evaluation the parameters **petask_<ID>_l_hard**, **petask_<ID>_l_soft**, **petask_<ID>_q_hard**, **petask_<ID>_q_soft** and **petask_<ID>_par**.

Read the *qsub*(1) man page for more information. There you can find a description of exported parameter names as well as corresponding values in the sections where command line switches are explained.

Please note that there are also pseudo parameters that describe characteristics of a job that are not passed as part of command line switches (like the command name or command arguments) or that describe the context or characteristics of the jobs submission itself. Find more information for those parameters in the next section.

PSEUDO PARAMETERS

CLIENT

The corresponding value for the *CLIENT* parameters is either 'qmaster' or the name of a submit client like 'qsub', 'qsh', 'qrsh', 'qlogin' and so on. This parameter value can't be changed by JSV instances. It will always be sent as part of a job verification.

CLIENT_COMMAND

Name of the submit client that wants to submit the job. Also in server JSVs the submit client name ('qsh', 'qrsh', 'qlogin' ...) is provided. *CLIENT_COMMAND* is a read-only parameter that cannot be adjusted by the JSV.

CMDARGS

Number of arguments which will be passed to the job script or command when the job execution is started. It will always be sent as part of a job verification. If no arguments should be passed to the job script or command it will have the value *0*. This parameter can be changed by JSV instances. If the value of *CMDARGS* is bigger than the number of available *CMDARG* parameters then the missing parameters will be automatically passed as empty parameters to the job script.

CMDNAME

Either the path to the script or the command name in case of binary submission. It will always be sent as part of a job verification.

CONTEXT

Either 'client' if the JSV which receives this param_command was started by a commandline client like qsub, qsh, ... or 'master' if it was started by the sge_qmaster process. It will always be sent as part of a job verification. Changing the value of this parameters is not possible within JSV instances.

GROUP

Defines Primary group of the user which tries to submit the job which should be verified. This parameter cannot be changed but is always sent as part of the verification process. The user name is passed as parameters with the name *USER*.

JOB_ID

Not available in the client context (see *CONTEXT*). Otherwise it contains the job number of the job which will be submitted to Grid Engine when the verification process is successful. *JOB_ID* is an optional parameter which can't be changed by JSV instances.

USER

Username of the user which tries to submit the job which should be verified. Cannot be changed but is always sent as part of the verification process. The group name is passed as parameter with the name *GROUP*

VERSION

VERSION will always be sent as part of a job verification process and it will always be the first parameter which is sent. It will contain a version number of the format .. In version 6.2u2 and higher the value will be '1.0'. The value of this parameter can't be changed.

SCRIPTSIZE

SCRIPTSIZE contains the size of the submitted job script. This parameter cannot be changed but it is always sent as part of the verification process.

EXAMPLE

Here is an example for the communication of a client with a JSV instance when following job is submitted:

```
qsub -pe p 3 -hard -l a=1,b=5 -soft -l q=all.q /jobs/sleeper.sh
```

Data in the first column is sent from the client/qmaster to the JSV instance. That data contained in the second column is sent from the JSV script to the client/qmaster. New line characters which terminate each line in the communication protocol are omitted.

client/qmaster -> JSV	JSV -> client/qmaster
START	
	SEND ENV STARTED
PARAM VERSION 1.0 PARAM CONTEXT client PARAM CLIENT qsub PARAM USER ernst PARAM GROUP staff PARAM CMDNAME /jobs/sleeper.sh	

client/qmaster -> JSV	JSV -> client/qmaster
PARAM CMDARGS 1	
PARAM CMDARG0 12	
PARAM l_hard a=1,b=5	
PARAM l_soft q=all.q	
PARAM M user@hostname	
PARAM N Sleeper	
PARAM o /dev/null	
PARAM pe_name pe1	
PARAM pe_min 3	
PARAM pe_max 3	
PARAM S /bin/sh	
BEGIN	
	RESULT STATE ACCEPT

SEE ALSO

sge_jsv_script_interface(3), *qsub*(1),

COPYRIGHT

sge_licensing

NAME

Licensing - Summary concerning licensing of Altair Grid Engine

GENERAL OVERVIEW

Altair Grid Engine 8.7.0 introduces a license mechanism that allows fine grained resource usage reporting which is linked to the Altair licensing services. License consumption information is available during the runtime of the main cluster component (sge_qmaster) for the current point in time and also for the past to the beginning where Altair Grid Engine 8.7.0 was installed or where an upgrade to that version was made.

The new license consumption reporting allows the usage of Altair network licenses maintained by the Altair licensing server. Altair Grid Engine consumes a certain amount of feature licenses dependent on the number of CPUs and GPUs available in the cluster.

The licensing functionality is implemented in all Altair Grid Engine components. Execution daemons (sge_execd) report resource availability and/or consumption. This information is collected in Altair Grid Engine's main component (sge_qmaster) and available for cluster managers via client commands (qconf, qstat, qhost).

Altair Grid Engine does check for license violations at regular time intervals every 90 seconds. The system notifies the manager of the cluster when violations occur and provides actions to solve license violations.

If for any reasons network licenses and/or installing the Altair license server are not an option for your configuration, please contact our support department to find a solution fulfilling your requirements.

LICENSED RESOURCES

Altair Grid Engine uses visible and usable CPU cores and GPUs that may be used for computational work in a cluster for license considerations. The availability of such resources consumes a corresponding entitlement of a Altair Grid Engine license.

The total available amount of CPU cores and GPUs is reported by each compute node as static load values with the name m_core and m_gpu. Static means that the reported resource consumes license entitlements beginning with installation until an execution node is uninstalled. When a compute node is not available (e.g during maintenance times) the Altair Grid Engine system will use the last reported values of m_core and m_gpu for license considerations unless all queues residing on that host are disabled manually by a manager of the cluster.

Reported m_core and m_gpu values might change during the lifetime of an execution node, when CPU sockets and/or GPUs are added, replaced or removed. In that case the change can be detected as soon as the corresponding execution daemon (see sge_execd(8)) reconnects to the main cluster component (see sge_qmaster(8)) when the first load report is sent. Changes in reported resource counts will be detected approximately 90 seconds after the execution daemon is restarted or within the configured load report time plus 90 seconds (see load_report_interval in sge_conf(5)) during runtime of a sge_execd. Reported load values can be made visible with different client commands (see qstat(1)/qhost(1) -F or qconf(1) -se). A resource usage check can also be manually triggered by using the qconf -tlv command.

The sge_qmaster process collects the load values of all execution nodes and accumulates the license count that will then be visible in form of License Usage Records (see qconf(1) -slur). Usage information collected in License Usage Records will also be accumulated and compared against the number of available feature licenses.

LICENSE USAGE RECORDS

License Usage Records describe resource consumption and shortage in a cluster over a range of time. The sum of all data sets shows available capacities or licensing violations.

Records are created automatically and they comprise the following describing attributes:

- *ID*: Identifier for one record
- *start_time* and *end_time*: Each record has a start and end time (visible as 64bit UNIX timestamp) that shows when and how long resources were reported to be available in the cluster. Whenever the total amount of resources changes in a cluster a new record is created. Usually the end time and start time of consecutive records match. Even in case of cluster downtimes (sge_qmaster(8) is inactive) the time gaps in license usage reporting will be closed as long as the downtime does not exceed 24 hours. If reported resources of hosts should not be considered during smaller maintenance windows then queues residing on a host have to be disabled manually and changes have to be reflected in a new license usage record (automatically created every 15 seconds) before cluster components should be shut down.
- *resource_consumption*: Shows consumed resources by a cluster. Directly after qmaster installation and before execution nodes are attached the *resource_consumption* will show the keyword **NONE**. As soon as execution nodes are installed this attribute will show one or multiple resources in the form of name/value pairs separated by commas. The names represent different resources whereas the numbers denote the total number of corresponding resource items available in the cluster. The following resources might be reported:
- *m_core_l*, *m_core_r*, *m_gpu_l*, *m_gpu_r*: Machine cores on premises (local), machine cores in the cloud (remote), GPUs on premises (local), GPUs in the cloud (remote). Reported cores and GPUs are considered to be **remote** if hosts objects or host configurations are correspondingly tagged. Find more information below how to tag hosts.
- *resource_shortage*: This attribute will show the keyword **NONE** as long as the installed license covers all available resources in a cluster. Otherwise it will show a comma separated list of name/value pairs of those resources that are not covered by the installed

license file. The same resources might be reported as with the *resource_consumption* attribute.
- *licenses_required*: Shows the number of Altair feature licenses required for the current available resources in a cluster.
- *licenses_checked_out*: Shows the number of Altair feature licenses checked out from the licensing server for the currently available resources in a cluster.
- *info*: Shows information about special states or occurrences of the licensing system. If everything is working as it should, this field shows the cluster to be licensed.

LICENSE TYPE

Prior to release AGE 2025.1.0 (8.10.0), a **Grid-Engine-Feature** license was provided via file-based licensing or an Altair License server.

With release AGE 2025.1.0, a new unified **Altair-Units** license is provided through an Altair License server. Altair-Units are not availble as File-based licensing. Existing Grid-Engine-Feature licenses continue to be accepted.

Each machine CPU will consume a weighted number of licenses. Each machine GPU will consume a different weighted number of licenses. The weights are different for the two types of licenses.

The license type is set via the **LICENSE_TYPE** value in the qmaster_params attribute. See sge_conf(5) for details.

LICENSING ACTIONS

Altair Grid Engine triggers specific actions when certain alarm levels are reached. There are four distinct alarm levels, namely info, warning, error and alert. Otherwise, Altair Grid Engine operates without any alarm level triggered when licensing is working without any problems.

Here is the description of actions that are taken by the Altair Grid Engine master component when an alarm level is reached.

- **info**: The info level informs about important, successful licensing events and is used primarily for debugging purposes. An info message is logged to Altair Grid Engine logging facility every 6 hours only if info messages are enabled there. No email message is sent.
- **warning**: The warning level occurs, when the licensing server reports a warning. This is usually the case, when the license file is about to expire. A warning message is logged to the Altair Grid Engine logging facility every 3 hours. An email is sent to the cluster administrator every 12 hours describing the warning as well.
- **error**: The error level occurs, when Altair Grid Engine was not able to acquire all licensing entitlements from the licensing server which are required to license all resources in the cluster. An error message is logged to the Altair Grid Engine logging facility every 20 minutes. Additionally, an email is sent to the cluster administrator every 3 hours describing the error.

- **alert**: The alert level is the most severe licensing alarm. It occurs, when the connection to the licensing server is lost. An error message is logged to the Altair Grid Engine logging facility every 10 minutes. An email is sent to the administrator of the cluster every hour describing the alert.

If the error or alert level was triggered and the last successful license validation is older than the granted grace period of 3 days, all hosts will enter the unlicensed state. Messages are logged and emails are sent as described above. Note that unlicensed hosts are excluded from scheduling. Jobs submitted to unlicensed hosts will enter pending state.

Jobs already running can always continue to run.

LICENSING ALGORITHM

The licensing algorithm is the instance in sge_qmaster responsible to report license violations and trigger corresponding actions. Depending on the severeness of the alarm level which the algorithm detects, it will:

- Cause logging messages to be written to the message file of qmaster
- Send emails to the admin or configured users, containing information about the license violations
- Set execution host queues to unlicensed state
- Exclude unlicensed resources from scheduling

On startup, the algorithm attempts to license all currently available resources in the cluster by checking out all required feature licenses. If not enough features licenses are available, the algorithm will get as many feature licenses as possible. If this operation fails, an initial grace period of 3 days will be granted. Within the initial grace period, the cluster continues to work normally but error log messages will be generated and emails will be sent. If this period expires without successfully checking out all feature licenses required, unlicensed hosts will be set to unlicensed state (L-state). It is possible to submit new jobs to unlicensed hosts but these jobs will not be scheduled until the host is licensed again. Unlicensed hosts are always excluded from scheduling. When all required feature licenses are checked out successfully, the cluster will start or continue to operate normally.

If an error or alert alarm level occurs, e.g. when the licensing server becomes unreachable or the license is expired, a grace period of 3 days is granted. Within this period, all cluster resources continue to work normally but error log messages are generated and emails are sent. Note that any occurance of a grace period will be logged in the License Usage Records. If this period expires without a successful feature license checkout, all hosts will enter the unlicensed state. Submission of new jobs will still be possible but these jobs will not be scheduled until at least one host becomes licensed.

The licensing algorithm will perform the following operations as a sequence:

- Accumulate all resources (m_gpu and m_core) for active hosts that cause license consumption (not in L or d-state).

- Calculate the feature license costs for those resources. One core requires 10, one GPU requires 100 feature licenses.
- Checkout the number of calculated feature licenses from the Altair licensing server and handle eventual errors.
- Check the state of the connection to the licensing server. Also, check for warnings, which usually occur, when the license is about to expire.
- Collect all alert levels and messages which may have occurred
- Check license usage. For every execution host, check if enough feature licenses are available. If enough licenses are available for the current host, keep the host enabled or re-enable the host, if it was in the unlicensed state. Contrariwise, if there are not enough licenses remaining, set or keep the unlicensed state of the host. Sum up the core and GPU resources of all hosts in the unlicensed state and save the values for the license resource shortage attribute, which is assigned to the License Usage Record in the next step.
- Update the license usage record. If any value of the License Usage Record has changed (see License Usage Record section), add a new entry to the list of License Usage Records.
- Handle any alert level and message, which may have occurred during the previous steps. Log an automated message to the Altair Grid Engine logging facility and send an email, if required.

This license verification algorithm will run once every 90 seconds. Note that you can trigger a manual verification run using the `qconf -tlv` command at any given time e.g. after an execution host was added to the cluster.

REQUIREMENTS

There are a couple of requirements that need to be fulfilled so that the licensing functionality of Altair Grid Engine can work properly:

- A working Altair License Management System installation with a proper license for the Altair Grid Engine cluster has to be up and running. The licensing server has to be introduced to the running instance of Altair Grid Engine by setting the proper qmaster configuration parameter named AGE_LICENSE_PATH. When installing the Qmaster component of Altair Grid Engine via the installation script, a dialogue will ask for the AGE_LICENSE_PATH aswell. See the Installation Guide and the “License Management System Guide” for further details on how to setup the Altair License Managment System.
- The execution daemons providing GPU information need to be able to query those devices. Access to GPU resources for jobs requires a working NVML (NVIDIA Management Library) installation and the NVML shared library to be loadable. Consider the documentation provided by NVIDIA for further details on how to setup NVML.

- A user specified path to the NVML shared library can be declared by adding **NVML_LIBRARY_PATH** to *execd_params*. See manual page *sge_conf(5)* for more information.
- If NVML is not available on the system, pcilib is used to find matching PCI devices, where the vendor id resolves to NVIDIA and the PCI device class is a 3D, XGA or VGA controller. Each of these devices is then reported as one GPU unit. Querying for PCI devices is available on Linux systems only. The pcilib dynamic library is a standard component on Linux systems.
- If Altair Grid Engine cannot find any of the aforementioned libraries, then zero GPU resources will be reported to the master component.

ADMINISTRATIVE COMMANDS

Display License Usage Over Time.

In a running cluster license usage will change over time, depending on the installed and active execution nodes and also depending on the hardware (CPU, GPU) installed on those machines.

License usage is collected over time and license usage records will be automatically created. Such license records can be shown with the command **qconf -slur**. Find more information concerning license usage records in the corresponding section above.

Trigger License Verification Manually

A license verification run can be triggered manually at any given time using the **qconf -tlv** command.

Setting licensing email

In order to determine the recipients of licensing related emails, the qmaster configuration parameter named LICENSING_MAIL has to be set. Its value has to be a valid email address or a list of email addresses separated by a colon.

If the LICENSING_MAIL parameter is not set, the administrator_mail configuration parameter is used instead if provided.

Enforce Reporting of Cloud Resources

Altair Grid Engine distinguishes between resources that are reported to be on premises or in the cloud. Depending on how the cluster is installed manual steps might be required to tag resources as cloud resources before they are recognized as such.

Reported cores or GPUs of a host are handled as cloud resources when one of the following conditions is met:

- The Altair Grid Engine host object contains in the *complex_values* the definition of a boolean complex named **tortuga** that is set to **true**. (Might require to create the boolean complex tortuga before use).
- The host configuration objects contains an *execd_param* where a **host_provider** parameter is set to a character sequence.

Disabling License Consumption for Specific Hosts and/or Resources

In case execution nodes fulfill a special role in a cluster (e.g where only transfer-queues are residing or when those hosts are used just to feed load information in a cluster) then the reported *m_code* and *m_gpu* values of such a node should not consume license entitlements. Also, in some situations, not all GPUs available on a system are appointed to consume feature licenses e.g. an integrated GPU in a CPU, which is not used.

Customers having such use cases should contact our Support organization to get instruction how to disable resource reporting.

AGERest interface

The AGERest interface allows to access information about license usage records.

ENVIRONMENT VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SEE ALSO

qconf(1), *qstat*(1), *qhost*(1), *qmod*(1), *sge_execd*(8), *sge_qmaster*(8)

COPYRIGHT

sge_nvidia(5)

NAME

sge_nvidia - Management of NVIDIA GPUs in Altair Grid Engine

DESCRIPTION

The sections below describe the setup and management of NVIDIA GPUs and Multi-Instance GPUs (MIG) in Altair Grid Engine.

OVERVIEW

Altair Grid Engine uses Resource Maps (complex with type `RSMAP`) to map physical GPUs and MIG to internal resources that can be used during the scheduling of jobs.

For an overview of Resource Maps and how to configure and request complexes see sge_resource_map(5) and sge_complex(5).

NVIDIA Management Library (NVML)

The NVIDIA Management Library (NVML) is used to query information about the GPUs of a host. The library is used to set the load value m_gpu (i.e. the number of installed physical GPUs) and to set CUDA specific load values that can be displayed via `qconf -se <hostname>`.

The execd parameter NVML_LIBRARY_PATH (see sge_conf(5)) can be used to specify a custom path to the NVML library. The user specified library will be preferred to the system library.

Note: The cuda_load_sensor that was shipped with previous versions of Altair Grid Engine used NVML to query and report GPU load values and is not needed anymore.

NVIDIA Data Center GPU Manager (DCGM)

NVIDIA Data Center GPU Manager (DCGM) can be used to check the health of the installed GPUs and to collect GPU specific usage values of jobs. Support for DCGM can be enabled by setting the execd parameter `DCGM_PORT` to the port DCGM uses on the host (the default port is `5555`). For a list of supported DCGM versions see the Release Notes.

Health Checks: DCGM performs basic health checks of GPUs. If DCGM reports a health problem for one of the installed GPUs, the load values “cuda.<gpu>.health” and

"cuda.<gpu>.health_message" will be shown in `qconf -se <hostname>`. A GPU with a health value != 0 will not be used for scheduling.

NVLink related errors are not reported by Altair Grid Engine.

Job Usage Statistics: DCGM can be used to collect GPU usage statistics of jobs. For more information see USAGE STATISTICS.

NOTE: Due to limitations in DCGM, GPU usage is currently not reported for jobs using a NVIDIA Multi-Instance GPU (MIG).

NVIDIA Multi-Instance GPU (MIG)

In Altair Grid Engine NVIDIA Multi-Instance GPUs can be used like any physical GPU by configuring a Resource Map id and mapping it to the MIG device.

However, there are some additional requirements:

- The cuda id has the format `<gpu_id>:<gpu_instance_id>`. Since a colon is not allowed as part of the Resource Map id name, the `cuda_id` has to be configured and `DOCKER_RESOLVE_CUDA_ID=true` (see sge_conf(5)) has to be set in order to use MIG devices for Docker jobs.
- To control access to MIG devices via cgroups access to the devices `/dev/nvidia[0-254]` and `/dev/nvidia-caps/nvidia-cap[0-254]` have to be blocked with the cgroups_param `devices`. Every Resource Map id representing a MIG has to allow access to its parent device and multiple devices in `/dev/nvidia-caps/` (see https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html).

GPU Load Values

If NVML and/or DCGM is available, the following GPU specific load values will be reported for hosts with NVIDIA GPUs. The load values are displayed in `qconf -se <hostname>`.

Value	NVML	DCGM	Description
cuda.devices	X		Number of installed NVIDIA devices
cuda.verstr	X		Installed driver version
cuda.<gpu>.affinity	X		Affinity between GPU <gpu> and the installed CPU(s) as reported by `nvidia-smi topo`
cuda.<gpu>.gpu_temp	X		GPU temperature
cuda.<gpu>.health		X	Health status of the GPU. Only reported when != 0
cuda.<gpu>.health_message		X	Health status message of the GPU
cuda.<gpu>.instances	X		Number of GPU instances (when MIG mode is enabled)
cuda.<gpu>.mem_total	X		Total memory of the GPU
cuda.<gpu>.mem_free	X		Free memory of the GPU
cuda.<gpu>.mem_used	X		Used memory of the GPU
cuda.<gpu>.mig_mode	X		MIG mode of the GPU (0 = disabled, 1 = enabled)
cuda.<gpu>.name	X		Name of the GPUs
cuda.<gpu>.path	X		Device path of the MIG (needed for cgroups)

Value	NVML	DCGM	Description
cuda.<gpu>.power_usage	X		Current power usage of the GPU
cuda.<gpu>.uuid	X		UUID of the GPU
cuda.<gpu>:<mig>.affinity	X		Affinity between the MIG and the installed CPU(s)
	X		as reported by `nvidia-smi topo`
cuda.<gpu>:<mig>.mem_total	X		Total memory of the MIG
cuda.<gpu>:<mig>.mem_free	X		Free memory of the MIG
cuda.<gpu>:<mig>.mem_used	X		Used memory of the MIG
cuda.<gpu>:<mig>.name	X		Name of the MIG
cuda.<gpu>:<mig>.path	X		Device path of the MIG (needed for cgroups)
cuda.<gpu>:<mig>.uuid	X		UUID of the MIG

NOTE: GPUs with MIG mode enabled cannot be used by CUDA applications and are therefore not considered for `cuda.devices`.

SETUP

Resource Maps

For an overview of Resource Maps and how to configure complexes with type `RSMAP` see sge_resource_map(5).

The following additional parameter can be used to map GPUs to Resource Map ids:

- `cuda_id`: Id of the GPU to which this id should be mapped (according to the output of `nvidia-smi`). The id will be used to export `CUDA_VISIBLE_DEVICES`.
- `uuid`: UUID of the GPU to which this id should be mapped. If configured, it will automatically be used instead of `cuda_id` for `CUDA_VISIBLE_DEVICES`.

Cgroups

Access to GPUs can be managed via cgroups to ensure that jobs are only able to access devices that were assigned to them.

If the cgroups_param **devices** is set to a list of device paths (separated by `|`), the cgroups devices subsystem/controller will be used to block access to GPUs in the list and jobs will only be able to access GPUs that were assigned to them via RSMAPs (see RMSAP parameter `device`)

The following configuration will block access to all NVIDIA GPUs and MIGs:

```
devices=/dev/nvidia[0-255]|/dev/nvidia-caps/nvidia-cap[0-255]
```

If a RSMAP with the device `/dev/nvidia0` is assigned to a job, the job will be able to access the GPU to which this device path belongs. The device paths for each GPU are shown in the `qconf -se <hostname>` output.

Jobs that do not request RSMAPs will not be able to access any of the installed GPUs.

Please note that /dev/nvidiactl is needed for jobs to access NVIDIA drivers, so access to this device should not be blocked.

For instructions on how to enable cgroups and how to set the cgroup_param **devices**, see sge_conf(5).

SCHEDULING

Requesting GPUs

For a overview of Resource Maps and how to request complexes with type `RSMAP` see sge_resource_map(5).

The following additional parameters can be requested for GPUs and will have an effect on the scheduling result:

- `affinity`: see GPU-CPU Affinity

GPU-CPU Affinity

If a job requests a GPU and **affinity**, it will automatically be bound to the CPU cores that have a good affinity to the assigned GPU (see `nvidia-smi topo`). This ensures that the data between the CPU and GPU is transferred in the fastest way possible.

The parameter supports two values:

- **1/true**: Affinity is requested as hard request
- **2**: Affinity is requested as a soft request

If affinity is requested as hard request, the job will not be scheduled unless there is a GPU/CPU pair with good affinity and enough free CPU cores available. If affinity is requested as a soft request, Altair Grid Engine will try to schedule the job with a GPU/CPU pair with good affinity, but schedule the job anyway without binding to any CPU cores, if not enough CPU cores are available and free. If less cores are needed the request can be combined with the **-binding** switch.

CUDA_VISIBLE_DEVICES

Altair Grid Engine can automatically export the environment variable `CUDA_VISIBLE_DEVICES` for jobs that request GPUs. For more information see sge_conf(5), `SET_CUDA_VISIBLE_DEVICES`.

NVIDIA Container Toolkit

The NVIDIA Container Toolkit can be used to allow access to GPUs within Docker containers. For more information about the toolkit and how to install it properly, see https://github.com/NVIDIA/nvidia-docker.

In order to use the NVIDIA Container Toolkit the switch `-xd "--runtime=nvidia"` must be specified in the **qsub** or **qrsh** command line. To select a specific GPU, the environment variable `NVIDIA_VISIBLE_DEVICES` must be set by using `-xd "--env NVIDIA_VISIBLE_DEVICES=<gpu_id>"`.

Altair Grid Engine also supports the Docker run option `--gpus` to select GPUs for a container. The switch accepts either `all` to select all GPUs on the host, any integer greater than zero to select a specific number of GPUs or the parameter `device` followed by a list of device ids to select specific GPUs, e.g. `-xd "--gpus=device=\"0,1\""`.

The Docker run option `--gpus` requires Docker API version 1.40 or newer. When `--gpus` is used, `--runtime=nvidia` and `--env NVIDIA_VISIBLE_DEVICES=<gpu_id>` can be ommited.

Placeholders in the submission command can be used to use GPUs that were assigned to a job via Resource Maps within a container. For more information about placeholders see submit(1). When the qmaster_param DOCKER_RESOLVE_CUDA_ID (see sge_conf(5)) is set to `true`, the `cuda_id` parameter of the assigned Resource Maps will be used to resolve placeholders.

USAGE STATISTICS

If enabled, DCGM can be used to collect GPU specific usage values that are displayed in the gpu_job_usage section of qstat -j <job_id> and qacct -j <job_id>.

GPU Job Usage

The following usage values are available. The values are displayed in qstat -j <job_id> in the format <hostname>(cuda.<gpu>.<usage_name>=<usage_value>).

Usage value	Description
boardLimitViolationTime	Number of seconds at reduced clocks due to the board's voltage limit
eccDoubleBit	Count of ECC double bit errors that occurred
endTime	Time when the job stopped using the GPU (in microseconds since 1970)
eccSingleBit	Count of ECC single bit errors that occurred
energyConsumed	Energy consumed by the gpu in Joules
lowUtilizationTime	Number of seconds reduced clocks due to low utilization
maxGpuMemoryUsed	Maximum amount of GPU memory that was used in MB
memoryClock_min	Minimum Memory clock in MHz
memoryClock_max	Maximum Memory clock in MHz
memoryClock_avg	Average Memory clock in MHz

Usage value	Description
memoryUtilization_min	Minimum GPU Memory Utilization in percent
memoryUtilization_max	Maximum GPU Memory Utilization in percent
memoryUtilization_avg	Average GPU Memory Utilization in percent
numOtherComputePids	Count of otherComputePids entries that are valid
numOtherGraphicsPids	Count of otherGraphicsPids entries that are valid
numXidCriticalErrors	Number of valid entries in xidCriticalErrorsTs
overallHealth	The overall health of the system
pcieReplays	Count of PCI-E replays that occurred
pcieRxBandwidth_min	Minimum PCI-E MB read from the GPU
pcieRxBandwidth_max	Maximum PCI-E MB read from the GPU
pcieRxBandwidth_avg	Average PCI-E MB read from the GPU
pcieTxBandwidth_min	Minimum PCI-E MB written to the GPU
pcieTxBandwidth_max	Maximum PCI-E MB written to the GPU
pcieTxBandwidth_avg	Average PCI-E MB written to the GPU
powerViolationTime	Number of seconds we were at reduced clocks due to power violation
processUtilization_sm	Process SM and Memory Utilization (in percent) Deprecated. Will be removed in one of the next versions of Altair Grid Engine.
processUtilization_mem	Process SM and Memory Utilization (in percent) Deprecated. Will be removed in one of the next versions of Altair Grid Engine.
reliabilityViolationTime	Number of seconds at reduced clocks due to reliability limit
smClock_min	Minimum SM clock in MHz
smClock_max	Maximum SM clock in MHz
smClock_avg	Average SM clock in MHz
smUtilization_min	Minimum GPU SM Utilization in percent
smUtilization_max	Maximum GPU SM Utilization in percent
smUtilization_avg	Average GPU SM Utilization in percent
startTime	Time when the job started using the GPU (in microseconds since 1970)
syncBoostTime	Number of seconds at reduced clocks due to sync boost
thermalViolationTime	Number of seconds reduced clocks due to thermal violation

Which values to report is defined via the global/local configuration attribute *gpu_job_usage*. The following values can be configured, the default value is *none*.

Setting	Description
none	No gpu usage values will be reported
all	All implemented usage values will be reported, see the table above
<variable list>	Comma or space separated list of variable names, without *cuda.<gpu>* e.g. *eccDoubleBit*, *maxGpuMemoryUsed*, *memoryClock_max*, …

GPU Job Accounting

GPU usage values can also be written to the accounting file and displayed in qacct -j <job_id>.

Which values to write to the accounting file is defined via the global/local configuration attribute *gpu_job_accounting*.

The default value for **gpu_job_accounting** is *none*. All settings that are valid for **gpu_job_usage** are also valid for **gpu_job_accounting**.

NOTE: **gpu_job_usage** and **gpu_job_accounting** can be configured individually. If **gpu_job_accounting** is set to "none", no gpu specific usage values will be written to the accounting file, regardless of the configuration of **gpu_job_usage**.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), qsub(1), complex(5), host(5), sge_job_class(5), sge_resource_map(5).

COPYRIGHT

path_map

NAME

path_map - Altair Grid Engine path mapping file format

DESCRIPTION

The path_map file is only necessary if a Windows (win-x86) host is part of the Altair Grid Engine cluster. It is used to define how UNIX paths shall be converted into Windows paths.

The path_map file contains a list of comma separated mappings between UNIX and the corresponding Windows paths. There are two columns in each line. In the left column, the UNIX path is listed, in the right column, the corresponding Windows path is listed. The columns are separated by a comma, which may be followed by a blank to improve readability.

The path_map file will be used only on Windows and by these Altair Grid Engine components: sge_execd(8), sge_shepherd(8), uge_js_service.exe(8) and all Altair Grid Engine client commands.

If these Altair Grid Engine components have to convert a path, they will scan the left column of the path_map file from top to bottom until they find a path that matches the beginning of the path to convert. This part will then be replaced by the Windows path from the right column of the same line. After this, all slashes in the path to convert will be replaced by backslashes. Because of this, it is not possible to use UNIX paths that contain backslashes for a Altair Grid Engine cluster that includes a Windows host.

A proper path_map file always contains a mapping with the pseudo path "/execd_spool_dir/win-x86/placeholder" in the left column. The same path appears in the configuration of a Windows execution host. It is just a placeholder that is necessary because it's not possible to configure Windows paths in the global or execution host configuration.

EXAMPLE

If the path to convert is "/home/jdoe/joboutputs/myjob.o17" and the path_map file looks like this:

```
/opt/UGE820, \\server\share\UGE820
/execd_spool_dir/win-x86/placeholder, c:\tmp\spool\UGE
/var/sgeCA, c:\tmp\var\sgeCA
/home, \\homeserver\homeshare
/tmp, c:\tmp
```

then Altair Grid Engine will scan over the lines in the *path_map* file until it finds in the left column a path that matches the beginning of the path to convert. In this case, this is the path "/home". It will be replaced in the path to convert by the content of the right column, resulting in the intermediate path "`\\homeserver\homeshare/jdoe/joboutputs/myjob.o17`". Then, all slashes in the path are replaced by backslashes, resulting in the final path "`\\homeserver\homeshare\jdoe\joboutputs\myjob.o17`".

SEE ALSO

sge_execd(8), sge_shepherd(8), uge_js_service.exe(8)

COPYRIGHT

sge_pe

NAME

sge_pe - Altair Grid Engine parallel environment configuration file format

DESCRIPTION

Parallel environments are parallel programming and runtime environments allowing for the execution of shared memory or distributed memory parallelized applications. Parallel environments usually require some kind of setup to be operational before starting parallel applications. Examples for common parallel environments are shared memory parallel operating systems and the distributed memory environments Parallel Virtual Machine (PVM) or Message Passing Interface (MPI).

sge_pe allows for the definition of interfaces to arbitrary parallel environments. Once a parallel environment is defined or modified with the **-ap** or **-mp** options to qconf(1) and linked with one or more queues via pe_list in sge_queue_conf(5) the environment can be requested for a job via the **-pe** switch to qsub(1) together with a request of a range for the number of parallel processes to be allocated by the job. Additional **-l** options may be used to specify the job requirement to further detail.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

PEs and cgroups

If cgroups is activated on a host and the individual cgroups-parameters are active, cgroups are set up for PE-jobs in the following way.

Note: The following notation is used for this chapter

name	abbr.
slave-task	ST
master-task	MT
control_slaves	cs
job_is_first_task	jift
daemon_forks_slaves	dfs
master_forks_slaves	mfs
+ masterl-request	+masterl

cpuset

PE-jobs with corebinding-requests behave in the following manner:

Below "gets all" means: All cores granted for this host will be combined and given to that process. E.g. On host A, the master-task and 4 slave-tasks are started, each granted one core. If the master-task "gets all" (see table below when this happens), then it will get 4 cores, namely the 4 cores used individually by each slave-task:

```
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/1.A/cpuset.cpus
0
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/2.A/cpuset.cpus
1
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/3.A/cpuset.cpus
2
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/4.A/cpuset.cpus
3
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/master/cpuset.cpus
0 1 2 3
```

The meaning of "gets unique part" can also be seen above. 4 cores are granted on host A, so each slave-task got 1 unique core of those.

PE-feature	slave-host	master-host
cs	each ST gets unique part	MT gets all, each ST gets unique part
cs + dfs	The only ST gets all	MT and only ST get all
cs + jift	each ST gets unique part	MT and each ST get unique part
cs + mfs	each ST gets unique part	MT gets all, no ST
cs + dfs + jift	each ST gets unique part	MT gets unique part, only ST gets the rest

Examples:

On each slave-host, if **control_slaves=true** and **daemon_forks_slaves=true**, only one slave-task will be started. This slave-task will get the combination of all granted cores on this host for this job (denoted above as "gets all"). E.g., for 2 slaves on this host, each granted 1 core:

```
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/1.examplehost/cpuset.cpus
0 1
```

If **daemon_forks_slaves=false**, each slave-task will get a sub-cgroups-directory, with its cores specified (denoted above as "gets unique part"). E.g.

```
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/1.examplehost/cpuset.cpus
```

```
0
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/2.examplehost/cpuset.cpus
1
```

For **control_slaves=true**, on the master-host, the master-task will get the combination of all granted cores E.g., for 2 slaves on this host, each granted 1 core:

```
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/master/cpuset.cpus
0 1
```

while the 2 slaves will also get these cores, but individually:

```
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/1.examplehost/cpuset.cpus
0
$ cat /sys/fs/cgroups/cpuset/UGE/30000001.1/2.examplehost/cpuset.cpus
1
```

Memory

Memory-limits behave according to the detailed description in **sge_diagnostics**.

devices

Devices behave similar to core-binding / cpuset, with the exception of the master-task. Below, “gets masterl” means either “gets nothing” if no masterl-request was done for a device, or it will get the device(s) requested with masterl.

PE-feature	slave-host	master-host
cs	each ST gets unique part	MT gets masterl, each ST gets unique part
cs + dfs	The only ST gets all	MT gets masterl, only ST get all
cs + jift	each ST gets unique part	MT gets masterl or unique part, each ST gets unique part
cs + mfs	each ST gets unique part	MT gets all, no ST
cs + dfs + jift	each ST gets unique part	MT gets masterl or unique part, only ST gets the rest

freezer

There are no sub-directories in the freezer-subsystem for PE-tasks. Only one directory for this job per host. Once the job gets suspended, all process IDs belonging to this job are written in this directory to freeze the job, i.e. PE-tasks are not frozen individually.

killing

This is used to ensure on the OS-level that all processes of this job finished.

For cgroup-enhanced killing, the cpuset subsystem is used. If the job requested corebinding, all PIDs are already contained within the job-directory and all task-subdirectories. If the job did not request binding, a cpuset-directory for this job is going to be created without any limits on core-usage and without sub-directories for each task. All PIDs of this job on this host will be collected inside this directory. If all processes of this job should be killed (e.g. if the job was deleted), the execution daemon will signal all processes until they all get removed from this directory by the OS.

FORMAT

The format of a sge_pe file is defined as follows:

pe_name

The name of the parallel environment as defined for pe_name in sge_types(1). To be used in the qsub(1) **-pe** switch.

slots

The number of parallel processes being allowed to run in total under the parallel environment concurrently. Type is number, valid values are 0 to 9999999.

used_slots

The number of currently occupied slots of this parallel environment. This is a read-only value and will only show with **-sp**!

bound_slots

The number of currently occupied slots of this parallel environment that got preempted, but are not yet available. This is a read-only value and will only show with **-sp**!

user_lists

A comma separated list of user access list names (see sge_access_list(5)). Each user contained in at least one of the enlisted access lists has access to the parallel environment. If the **user_lists** parameter is set to NONE (the default) any user has access being not explicitly excluded via the **xuser_lists** parameter described below. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the parallel environment.

xuser_lists

The **xuser_lists** parameter contains a comma separated list of so called user access lists as described in sge_access_list(5). Each user contained in at least one of the enlisted access lists is not allowed to access the parallel environment. If the **xuser_lists** parameter is set to NONE (the default) any user has access. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the parallel environment.

start_proc_args

The invocation command line of a start-up procedure for the parallel environment or the keyword NONE if no startup-script should be executed. The start-up procedure is invoked by sge_shepherd(8) for the master task of a parallel job after a possibly configured prolog (see sge_conf(5)) and prior to executing the job script. Its purpose is to setup the parallel environment correspondingly to its needs. An optional prefix "user@" specifies the user under which this procedure is to be started. The standard output of the start-up procedure is redirected to the file REQUEST.po_JID in the job's working directory (see qsub(1)), with REQUEST being the name of the job as displayed by qstat(1) and JID being the job's identification number. Likewise, the standard error output is redirected to REQUEST.pe_JID
Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration.
The following special variables being expanded at runtime can be used (besides any other strings which have to be interpreted by the start and stop procedures) to constitute a command line:

Value	Description
$pe_hostfile	The pathname of a file containing a detailed description of the layout of the parallel environment to be setup by the start-up procedure. See sge_pe_hostfile(5).
$host	The name of the host on which the start-up or stop procedures are started.
$job_owner	The user name of the job owner.
$job_id	Altair Grid Engine's unique job identification number.
$job_name	The name of the job.
$pe	The name of the parallel environment in use.
$pe_slots	Number of slots granted for the job.
$processors	The **processors** string as contained in the queue configuration (see *sge_queue_conf*(5)) of the master queue (the queue in which the start-up and stop procedures are started).
$queue	The cluster queue of the master queue instance.

stop_proc_args

The invocation command line of a shutdown procedure for the parallel environment or the keyword NONE if no shutdown procedure should be executed. The shutdown procedure is invoked by sge_shepherd(8) after the job script has finished, but before a possibly config-

ured epilog (see sge_conf(5)) is started. Its purpose is to stop the parallel environment and to remove it from all participating systems. An optional prefix "user@" specifies the user under which this procedure is to be started. The standard output of the stop procedure is also redirected to the file REQUEST.po_JID in the job's working directory (see qsub(1)), with REQUEST being the name of the job as displayed by qstat(1) and JID being the job's identification number. Likewise, the standard error output is redirected to REQUEST.pe_JID_
Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration.
The same special variables as for **start_proc_args** can be used to constitute a command line.

per_pe_task_prolog

The invocation command line of a startup procedure for each slave task of a tightly integrated parallel job or the keyword NONE if no startup procedure should be executed. The startup procedure is invoked by sge_shepherd(8) for each single slave task of a tightly integrated parallel job that is started using "qrsh -inherit" prior to executing the slave task script. Its purpose is to do setup tasks specific for this slave task, which is mainly useful if the slave task is started in Docker container. An optional prefix "user@" specifies the user under which this procedure is to be started. The standard output of the start-up procedure is redirected to the file REQUEST.po_JID in the job's working directory (see qrsh(1)), with REQUEST being the name of the job as displayed by qstat(1) and JID being the job's identification number. Likewise, the standard error output is redirected to REQUEST.pe_JID
Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration.
The same special variables as for **start_proc_args** can be used to constitute a command line.

per_pe_task_epilog

The invocation command line of a shutdown procedure for each slave task of a tightly integrated parallel job or the keyword NONE if no shutdown procedure should be executed. The shutdown procedure is invoked by sge_shepherd(8) for each single slave task of a tightly integrated parallel job that is started using "qrsh -inherit" after the end of the slave task script. Its purpose is to cleanup whatever there was configured in the corresponding **per_pe_task_prolog** for this slave task. An optional prefix "user@" specifies the user under which this procedure is to be started. The standard output of the shutdown procedure is redirected to the file REQUEST.po_JID in the job's working directory (see qrsh(1)), with REQUEST being the name of the job as displayed by qstat(1) and JID being the job's identification number. Likewise, the standard error output is redirected to REQUEST.pe_JID
Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration.
The same special variables as for **start_proc_args** can be used to constitute a command line.

allocation_rule

The allocation rule is interpreted by the scheduler thread and helps the scheduler to decide how to distribute parallel processes among the available machines. If, for instance, a parallel environment is built for shared memory applications only, all parallel processes have to be assigned to a single machine, no matter how much suitable machines are available. If, however, the parallel environment follows the distributed memory paradigm, an even distribution of processes among machines may be favorable.
The allocation rule always refers to hosts, not to queues. So if there are specific queues requested for the parallel job, e.g. using the "-q" or "-masterq" switch (see qsub(1)) the tasks of the job might get distributed over several queues, but the sum of tasks on one host will be always the one defined by the allocation rule.
The current version of the scheduler only understands the following allocation rules:

- : An integer number that sets the number of processes per host. A value of 1 limits processes to 1 process per host. When the "-pe", Parallel Environment option is specified (see qsub(1)) the number of tasks specified with the "-pe" option must divide evenly into otherwise the job will not be scheduled. If a master queue is requested using the "-masterq" option and a list of queues are specified with "-q" for the slave tasks and the master queue is not a member of the slave queue list specified by "-q" then the job will be scheduled if and only if the number of tasks specified in "-q" minus one divides evenly into .

If the special denominator **$pe_slots** is used, the full range of processes as specified with the qsub(1) **-pe** switch has to be allocated on a single host (no matter which value belonging to the range is finally chosen for the job to be allocated).

- *$fill_up*: Starting from the best suitable host/queue, all available slots are allocated. Further hosts and queues are "filled up" as long as a job still requires slots for parallel tasks.
- *$round_robin*: From all suitable hosts a single slot is allocated until all tasks requested by the parallel job are dispatched. If more tasks are requested than suitable hosts are found, allocation starts again from the first host. The allocation scheme walks through suitable hosts in a best-suitable-first order.

control_slaves

This parameter can be set to TRUE or FALSE (the default). It indicates whether Altair Grid Engine is the creator of the slave tasks of a parallel application via sge_execd(8) and sge_shepherd(8) and thus has full control over all processes in a parallel application, which enables capabilities such as resource limitation and correct accounting. However, to gain control over the slave tasks of a parallel application, a sophisticated PE interface is required, which works closely together with Altair Grid Engine facilities. Such PE interfaces are available through your local Altair Grid Engine support office.

Please set the control_slaves parameter to false for all other PE interfaces.

job_is_first_task

The **job_is_first_task** parameter can be set to TRUE or FALSE. A value of TRUE indicates that the Altair Grid Engine job script already contains one of the tasks of the parallel application, while a value of FALSE indicates that the job script (and its child processes) is not part of the parallel program.

This affects the number of pe tasks which can be started by the job script on the master host via qrsh -inherit: With **job_is_first_task** set to FALSE, the number of qrsh -inherit calls to the master host equals the number of slots the job was granted on the master host. With **job_is_first_task** set to TRUE, the number of qrsh -inherit calls to the master host is one less than the number of granted slots on the master host as the job script is considered handling the work of one task.

With **job_is_first_task** being set to FALSE in the parallel environment granted to the job, the job script (and its child processes) are considered to just start the parallel program, are not being part of the parallel program itself, are not considered to consume a significant amount of cpu time, memory and other resources and therefore no slot is granted to the job script and global resource requests are not applied to the parallel program itself, only task specific resource requests for parallel task 0 are. Because of this the job gets one task more than requested while the number of slots occupied by the job is equal to the number of requested tasks. Still it is possible to define separate resource requests for each single task, so with **job_is_first_task** being set to FALSE, the ID of the last PE task is equal to the number of requested PE tasks for this job. With **job_is_first_task** being set to TRUE, the ID of the last PE task is one less as than the number of requested PE tasks for this job.

Examples:
These submit command lines request a special queue for their last PE task:

```
$ qsub -pe jift_false.pe 4 -petask 4 -q last_task.q job.sh
$ qsub -pe jift_true.pe 4 -petask 3 -q last_task.q job.sh
```

Because this can be confusing and a source of errors, setting **job_is_first_task** to FALSE should be avoided. Instead, for the whole job the needed number of tasks should be requested and the resources should be requested accordingly.

urgency_slots

For pending jobs with a slot range PE request the number of slots is not determined. This setting specifies the method to be used by Altair Grid Engine to assess the number of slots such jobs might finally get.

The assumed slot allocation has a meaning when determining the resource-request-based priority contribution for numeric resources as described in sge_priority(5) and is displayed when qstat(1) is run without **-g t** option.

The following methods are supported:

- : The specified integer number is directly used as prospective slot amount.

- *min*: The slot range minimum is used as prospective slot amount. If no lower bound is specified with the range 1 is assumed.
- *max*: The of the slot range maximum is used as prospective slot amount. If no upper bound is specified with the range the absolute maximum possible due to the PE's **slots** setting is assumed.
- *avg*: The average of all numbers occurring within the job's PE range request is assumed.

accounting_summary

This parameter is only checked if **control_slaves** (see above) is set to TRUE and thus Altair Grid Engine is the creator of the slave tasks of a parallel application via sge_execd(8) and sge_shepherd(8). In this case, accounting information is available for every single slave task started by Altair Grid Engine.

The **accounting_summary** parameter can be set to TRUE or FALSE. A value of TRUE indicates that only a single accounting record is written to the sge_accounting(5) file, containing the accounting summary of the whole job including all slave tasks, while a value of FALSE indicates an individual sge_accounting(5) record is written for every slave task, as well as for the master task.**
Note:** When running tightly integrated jobs with SHARETREE_RESERVED_USAGE set, and with having *accounting_summary* enabled in the parallel environment, reserved usage will only be reported by the master task of the parallel job. No per parallel task usage records will be sent from execd to qmaster, which can significantly reduce load on qmaster when running large tightly integrated parallel jobs.

daemon_forks_slaves

This parameter is only checked if **control_slaves** (see above) is set to TRUE and thus Altair Grid Engine is the creator of the slave tasks of a parallel application via sge_execd(8) and sge_shepherd(8).

The **daemon_forks_slaves** parameter defines if every task of a tightly integrated parallel job gets started individually via **qrsh -inherit** (default value FALSE, e.g. used for mpich integration) or if a single daemon is started via **qrsh -inherit** on every slave host which forks the slave tasks (value TRUE, e.g. used for openmpi or lam integration).

With **daemon_forks_slaves** set to TRUE only a single task (the daemon) may get started per slave host, all limits set for this task are multiplied by the number of slots granted on the host.

master_forks_slaves

The **master_forks_slaves** parameter can be set to TRUE if the master task (e.g. mpirun called in the job script) starts tasks running on the master host via fork/exec instead of starting them via **qrsh -inherit.**

With **master_forks_slaves** set to TRUE all limits set for the master task (the job script) will be increased by the slave task limit multiplied by the number of slots granted on the host. No further tasks can be started on the master host via **qrsh -inherit**.

RESTRICTIONS

Note, that the functionality of the start-up, shutdown and signaling procedures remains the full responsibility of the administrator configuring the parallel environment. Altair Grid Engine will just invoke these procedures and evaluate their exit status. If the procedures do not perform their tasks properly or if the parallel environment or the parallel application behave unexpectedly, Altair Grid Engine has no means to detect this.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), qdel(1), qmod(1), qsub(1), sge_access_list(5), sge_qmaster(8), sge_shepherd(8).

COPYRIGHT

sge_pe_hostfile

NAME

sge_pe_hostfile(5) - Altair Grid Engine pe_hostfile file format

DESCRIPTION

When a parallel job is started in a Altair Grid Engine system the sge_execd(8) writes a pe_hostfile containing a detailed description of the layout of the parallel job.

The path to the pe_hostfile is passed to the job via environment variable PE_HOSTFILE.

Each line of the file refers to a host / queue instance on which parallel processes are to be run and contains 4 entries:

1. the hostname
2. the number of parallel processes to be run on the host
3. the name of the queue instance (cluster queue on host)
4. optionally a processor range to be used in case core binding
    - if no core binding is used this column contains “<NULL>”
    - in order for binding to be reflected in the pe_hostfile binding_instance must be “pe”, see submit(1)
    - processor range is a colon (:) separated list of cores, a core is printed as “<socket>,<core>”, e.g. “0,1” means the second core on the first socket

EXAMPLES

without core binding

```
bash-4.2$ qrsh -pe openmpi 4 cat '$PE_HOSTFILE'
host1 1 all.q@host1 <NULL>
host2 1 all.q@host2 <NULL>
host3 1 all.q@host3 <NULL>
host4 1 all.q@host4 <NULL>
```

with core binding

```
bash-4.2$ qrsh -pe openmpi 4 -binding pe linear:1 cat '$PE_HOSTFILE'
host1 1 all.q@host1 0,0
host2 1 all.q@host2 0,0
```

```
host3 1 all.q@host3 0,0
host4 1 all.q@host4 0,0

bash-4.2$ qrsh -pe openmpi 4 -binding pe linear:2 cat '$PE_HOSTFILE'
host1 2 all.q@host1 0,0:0,1:0,2:0,3
host2 2 all.q@host2 0,0:0,1:0,2:0,3

bash-4.2$ qrsh -pe openmpi 4 -binding pe one_socket_per_task:2 cat '$PE_HOSTFILE'
host1 2 all.q@host1 0,0:0,1:1,0:1,1
host2 2 all.q@host2 0,0:0,1:1,0:1,1
```

SEE ALSO

sge_intro(1), sge_types(1), submit(1), sge_pe(5), sge_execd(8).

COPYRIGHT

sge_preemption

NAME

Preemption - Manual, Semi-Automatic and Automatic Preemption in Altair Grid Engine

DESCRIPTION

Altair Grid Engine clusters can cope with different types of workloads. The configuration of the scheduler component defines the way how to handle different workloads in the daily operation. Various policies can be combined to reflect the requirements.

In previous versions of Grid Engine enforcing policies sometimes was difficult especially when high priority jobs would require resources of lower priority jobs that already bind resources like slots, memory or licenses. In such cases it was required to use slot-wise suspend on subordinate to make such resources available or reservation and advance reservation functionality could be used to reserve resources for such high priority jobs before they drop in.

Altair Grid Engine 8.3 (and above) additionally provide the possibility to enforce configured policies when required resources are already in use. This can be done through preemption. This document describes preemptive scheduling as an addition to the Altair Grid Engine job handling and scheduling that makes it possible to more closely follow the goals defined by the policies and if necessary enforce them.

TERMS

Following paragraphs describe a couple of terms that are used throughout this document.

Jobs which have high priority based on the configured policies can get the role of an *preemption consumer* that can cause a *preemption action* to be performed for one or more running jobs that have the role of a *preemption provider*. In general all those running jobs are considered as *preemption provider* where the priority is smaller than that of the *preemption consumer*.

There are different *preemption actions* available in Altair Grid Engine. What all of them have in common is that they will make all or a subset of the bound resources of a *preemption provider* available so that they can be used by one or more *preemption consumer*. Different *preemption actions* differ in the way how bound resources are freed and how the Altair Grid Engine system will make the bound resources available.

Preemption actions can be executed by Altair Grid Engine due to three different *preemption triggers*. A *preemption trigger* will define the time and has an influence on the chosen *preemption action* that is performed. In general *preemption trigger* can be *manual*, *semi-automatic* or *automatic*.

A *preemption consumer* that consumes resources that got available through triggering a *preemption action* has the role on an *preemptor* whereas those jobs that get forced to free resources are considered as *preemptee*.

Please note: Within Altair Grid Engine 8.3 *manual preemption* is implemented. *semi-automatic* or *automatic* trigger will follow with upcoming releases.

PREEMPTIVE TRIGGER AND ACTIONS

Altair Grid Engine 8.3 provides six different preemption actions to preempt a job. With manual preemption the user/manager has to choose which of the available preemptive actions should be used to trigger preemption of a job. With semi-automatic and automatic preemption mechanisms (available with future versions of Altair Grid Engine) either the system configuration or the Altair Grid Engine scheduler decides automatically which preemption action will be taken to release resources.

The six preemptive actions differ in the way which of the resources will be available for other jobs after the preemptee got preempted. Some of those actions have restrictions on which job types they can be applied as well as who is allowed to trigger them. The actions differ also in the way how they treat the processes that are executed on behalf of a job that gets preempted.

Within Altair Grid Engine all preemptive actions are represented by single capital letter (**T**, **R**, **C**, **P**, **N** or **S**) that is either passed to a command, specified in a configuration object or that is shown in command output to show the internal state of a job.

Some of the preemptive actions trigger the *suspend_method* that might be defined in the queue where the preemptee is executed. To be able to distinguish different preemption actions within the *suspend_method* an optional argument named *$action* might be used as pseudo argument when the method is defined. That argument will be expanded to the corresponding letter that represents the preemptive action during runtime.

(**T**)erminate Action: The preemptee will be terminated. As soon as all underlying processes are terminated all resources that were bound by that preemptee will be reported as free. The T-action can be applied to any job. Users can apply it only to own jobs.

(**C**)heckpoint Action: The preemptee will be checkpointed. As soon as a checkpoint is written and all underlying processes are terminated all bound resources will be reported as available and the job will be rescheduled. This preemption action can only be applied to checkpointing jobs where a checkpointing environment was specified during submission of this job.

(**R**)erun Action: The preempted job will be rescheduled. As soon as all underlying processes are terminated all bound resources will be reported as available. Managers can enforce the rerun of jobs even if those jobs are not tagged as rerun-able on the job or queue level.

(**P**)reemption Action: The preemptee will be preempted. Preempted means that the configured *queue-suspend* method (*$action* set to *P*) will be executed that might trigger additional operations to notify the processes about the upcoming preemption so that those processes can release bound resources by itself. After that the processes are suspended and all consumable resources, where the attribute available-after-preemption (aapre) is set to true, are reported as free. Not-available-after-preemption resources are still reported to

be bound by the preempted job. The preemption action can be applied to all preemption providers whereas users can only preempt own jobs.

e(**N**)hanced Suspend Action: Similar to the preempt action the queue *suspend_method* (*$action* set to "*N*") will be triggered before the preemptee gets suspended. Only non-memory-based consumables (including LO-managed license resources) are reported as free when the processes are suspended. Memory-based consumables that are available-after-preemption and also not-available-after-preemption consumables will still be reported as bound by the enhanced suspended job. This preemption action can be applied to all preemption providers. Users can only preempt own jobs.

(**S**)uspend Action: Similar to the preempt action the triggered method will be the *suspend_method* (*$action* set to "*S*") before the preemptee gets suspended. Only consumed slots (and LO-managed license resources) will be available after suspension. All other resources, independent if they are tagged as available-after-preemption or not-available-after-preemption in the complex configuration, will be reported as still in use. This preemption action can be applied to all preemption providers. Users can only preempt own jobs.

Which of the six preemptive action should be chosen to manually preempt a job? If a job is checkpointable then it should be the **C**-action. Here all consumed resources of the preemptee will be available for higher priority jobs. The preemptee can continue its work at that point where the last checkpoint was written when it is restarted.

Although also the **T**-action and the **R**-action provide the full set of resources but they should be seen as the last resort when no less disruptive preemptive actions can be applied. Reason for this is that the computational work of the preemptee up to the point in time where the preemptee is rescheduled or terminated might get completely lost which would be a waste of resources.

From the Altair Grid Engine perspective also the P-action makes all bound resources (slots + memory + other consumable resources where *aapree* of the complex is set to *true*) available for higher priority jobs. But this is only correct if the machine has enough swap space configured so that the underlying OS is able to move consumed physical memory pages of the suspended processes into that swap space and also when the application either releases consumed resources (like software licenses, special devices, ...) automatically or when a *suspend_method* can be configured to trigger the release of those resources. The **N**-action can be used for jobs that run on hosts without or with little configured swap space. It will make only non-memory-based consumables available (slots + other consumable resources where *aapree* of the complex is set to *true*).

If jobs either do not use other resources (like software licenses, special devices, ...) and when memory consumption is not of interest in the cluster, then the **S**-action can be chosen. It is the simplest preemptive action that provides slots (and LO-licenses) only after preemption. Please note that the S-action and S-state of jobs is different from the s-state of a job (triggered via *qmod* -s command). A regularely suspended job does not release slots of that job. Those slots are blocked by the manually suspended job.

The **P** and **N**-action will make consumable resources of preemptees available for higher priority jobs. This will be done automatically for all preconfigured consumable resources in a cluster. For those complexes the
available-after-preemption-attribute (*aapre*) is set to *YES*. Managers of a cluster can change

this for predefined complexes. They also have to decide if a self-defined resource gets available after preemption. For Resources that should be ignored by the preemptive scheduling functionality the *aapre*-attribute can be set to *NO*.

Please note that the resource set for each explained preemptive action defines the maximum set of resources that might get available through that preemption action. Additional scheduling parameters (like *prioritize_preemptees* or *preemptees_keep_resources* that are further explained below) might reduce the resource set that get available through preemption to a subset (only those resources that are demanded by a specified *preemption_consumer*) of the maximum set.

MANUAL PREEMPTION

Manual preemption can be triggered with the *qmod* command in combination with the *p*-switch. The *p*-switch expects one job ID of a *preemption_consumer* followed by one or multiple job ID's or job names of *preemption_provider*. As last argument the command allows to specify a character representing one of the six *preemptive_actions*. This last argument is optional. *P*-action will be used as default if the argument is omitted.

Syntax:

```
qmod [-f] -p <preemption_consumer>
          <preemption_provider> [<preemption_provider> ...]
          [<preemption_action>]

<preemption_consumer> := <job_ID> .
<preemption_provider> := <job_ID> | <job_name> .
<preemption_action>   := "P" | "N" | "S" | "C" | "R" | "T" .
```

The manual preemption request will only be accepted if it is valid. Manual preemption request will be rejected when:

- Resource reservation is disabled in the cluster.
- Preemption is disabled in the cluster.
- *preemption_consumer* has no reservation request.
- At least one specified *preemption_provider* is not running.
- **C**-action is requested but there is at least one *preemption_provider* that is not checkpointable.
- **R**-action is requested but there is at least one *preemption_provider* that is neither tagged as rerunnable nor the queue where the job is running is a rerunnable queue. (Manager can enforce the R-action in combination with the *f*-switch).

Manual preemption requests are not immediately executed after they have been accepted by the system. The Altair Grid Engine scheduler is responsible to trigger manual preemption during the next scheduling run. Preemption will only be triggered if the resources will not otherwise be available to start the preemption consumer within a configurable time frame (see *preemption_distance* below). If enough resources are available or when the scheduler

sees that they will be available in near future then the manual preemption request will be ignored.

Please note that resources that get available through preemption are only reserved for the specified *preemption_consumer* if there are no other jobs of higher priority that also demands those resources. If there are jobs of higher priority then those jobs will get the resources and the specified *preemption_consumer* might stay in pending state till either the higher priority jobs leaves the system or another manual preemption request is triggered.

Preemptees will automatically trigger a reservation of those resources that they have lost due to preemption. This means that they can be reactivated as soon as they are eligible due to their priority and as soon as the missing resources get available. There is no dependency between a preemptor and the preemptees. All or a subset of preemptees might get restarted even if the preemptor is still running if demanded resources are added to the cluster or get available due to the job end of other jobs.

Preemtees will have the jobs state **P**, **N** or **S** (shown in the *qstat* output or *qmon* dialogs) depending on the corresponding preemption action that was triggered. Those jobs, as well as preemptees that get rescheduled due to the **R**-action, will appear as pending jobs even if they still hold some resources. Please note that regularly suspended jobs (in **s**-state due to *qmod* -s) still consume all resources and therefore block the queue slots for other jobs. *qstat* -j command can be used to see which resources are still bound by preemptees.

PREEMPTION CONFIGURATION

The following scheduling configuration parameters are available to influence the preemptive scheduling as well as the preemption behaviour of the Altair Grid Engine cluster:

max_preemptees: The maximum number of preemptees in the cluster. As preempted jobs might hold some resources (e.g memory) and through the *preemptees_keep_resources* parameter might even hold most of their resources a high number of preemptees can significantly impact cluster operation. Limiting the number of preemptees will limit the amount of held but unused resources.

prioritize_preemptees: By setting this parameter to *true* or *1* preemptees get a reservation before the regular scheduling is done. This can be used to ensure that preemptees get restarted again at latest when the preemptor finishes, unless resources required by the preemptee are still held by jobs which got backfilled. *prioritize_preemptees* in combination with disabling of backfilling provides a guarantee that preemptees get restarted at least when the preemptor finishes, at the expense of lower cluster utilization.

preemptees_keep_resources: When a job gets preempted only those resources will get freed which will be consumed by the preemptor. This prevents resources of a preemptee from getting consumed by other jobs. *prioritize_preemptees* and *preemptees_keep_resources* in combination provide a guarantee that preemptees get restarted at latest when the preemptor finishes, at the expense of a waste of resources and bad cluster utilization. Exception: Licenses managed through LO and a license manager cannot be held by a preemptee. As the preemptee processes will be suspended the license manager might assume the license to be free which will lead to the license be consumed by a different job. When the preemptee processes get unsuspended again a license query would fail if the license is held.

preemption_distance: A preemption will only be triggered if the resource reservation that could be done for a job is farther in the future than the given time interval (hh:mm:ss, default 00:15:00). Reservation can be disabled by setting the value to 00:00:00. Reservation will also be omitted if preemption of jobs is forced by a manager manually using (via *qmod* -f -p ...).

PREEMPTION IN COMBINATION WITH LICENSE ORCHESTRATOR

License complexes that are reported by License Orchestrator are automatically defined as available-after-preemption (*aapre* is set to *YES*). This means that if a Altair Grid Engine job that consumes a LO-license resource gets preempted, then this will automatically cause preemption of the corresponding LO-license request. The license will be freed and is then available for other jobs.

Manual preemption triggered in one Altair Grid Engine cluster does not provide a guarantee that the specified preemption consumer (or even a different job within the same Altair Grid Engine cluster) will get the released resources. The decision which cluster will get the released resource depends completely on the setup of the License Orchestrator cluster. Consequently it might happen that a license resource that gets available through preemption in one cluster will be given to a job in a different cluster if the final priority of the job/cluster is higher than that of the specified preemption consumer.

COMMON USE CASES

A) License consumable (without LO)

Scenario: There is a license-consumable defined that has a maximum capacity and multiple jobs compete for those by requesting one or multiple of those licenses.

Complex definition:

```
$ qconf -sc
...
license  lic  INT  <=  YES  YES  0  0  YES
...
```

The last *YES* defines the value of *aapre*. This means that the license resource will be available after preemption.

License capacity is defined on global level:

```
$ qconf -se global
...
complex_values   license=2
```

When now two jobs are submitted into the cluster then both licenses can be consumed by the jobs.

```
$ qsub -l lic=1 -b y -l h_rt=1:00:00 sleep 3600
$ qsub -l lic=1 -b y -l h_rt=1:00:00 sleep 3600
...

$ qstat -F lic
...
all.q@rgbtest                  BIPC  0/1/60   lx-amd64
       gc:license=0
3000000005 0.55476 sleep      user         r
----------------------------------------------------------------------------------
all.q@waikiki                  BIPC  0/1/10         0.00     lx-amd64
       gc:license=0
3000000004 0.55476 sleep      user         r     04/02/2015 12:32:54     1
```

Submission of a higher priority job requesting 2 licenses and resource reservation:

```
$ qsub -p 100 -R y -l lic=2 -b y -l h_rt=1:00:00 sleep 3600
```

The high priority job stays pending, it will get a reservation, but only after both lower priority jobs are expected to finish:

```
$ qstat -j 3000000006
...
reservation       1:    from 04/02/2015 13:33:54 to 04/02/2015 14:34:54
                        all.q@hookipa: 1
```

We want the high priority job to get started immediately, therefore we trigger a manual preemption of the two lower priority jobs:

```
$ qmod -p 3000000006 3000000004 3000000005 P
Accepted preemption request for preemptor candidate 3000000006
```

The lower priority jobs get preempted, the high priority job can start:

```
$ qstat
job-ID     prior   name  user state submit/start at     queue     jclass slots ja-task-ID
-----------------------------------------------------------------------------------------
3000000006 0.60361 sleep joga r 04/02/2015 12:37:50 all.q@waikiki        1
3000000004 0.55476 sleep joga P 04/02/2015 12:32:54                      1
3000000005 0.55476 sleep joga P 04/02/2015 12:32:54                      1
```

Resources which have been preempted are shown in qstat -j . In order for the preemptees to be able to resume work as soon as possible, preempted jobs get a resource reservation for the resources they released, e.g.

```
$ qstat -j 3000000004
...
preempted   1: license, slots
usage       1: wallclock=00:04:45, cpu=00:00:00, mem=0.00015 GBs, io=0.00009,
               vmem=19.414M, maxvmem=19.414M
reservation 1: from 04/02/2015 13:38:50 to 05/09/2151 19:07:05
               all.q@waikiki: 1
```

B) License managed via LO that is connected to two different Altair Grid Engine clusters

Scenario: There is a license-consumable defined that has a maximum capacity and multiple jobs from two different connected Altair Grid Engine clusters (named A and B) compete for those by requesting one or multiple of those licenses.

TODO

SEE ALSO

sge_intro(1)

COPYRIGHT

sge_priority

NAME

sge_priority - Altair Grid Engine job priorities

DESCRIPTION

Altair Grid Engine provide means for controlling job dispatch and run-time priorities. The dispatch priority indicates the importance of pending jobs compared with each other and determines the order in which Altair Grid Engine dispatches jobs to queue instances. The run-time priority determines the CPU allocation that the operating system assigns to jobs.

JOBS DISPATCH PRIORITY

A job's dispatch priority is affected by a number of factors:

- the identity of the submitting user
- the project under which the job is submitted (or alternatively, the default project of the submitting user)
- any resources requested by the job
- the job's submit time
- the job's initiation deadline time (if specified)
- the -p priority specified for the job (also known as the POSIX priority "pprio")

The effect of each of these is governed by the overall policy setup, which is split into three top-level contributions. Each of these is configured through the *sge_sched_conf*(5) parameters **weight_priority**, **weight_ticket** and **weight_urgency**. These three parameters control to what degree POSIX priority, ticket policy, and urgency policy are in effect. To facilitate interpretation, the raw priorities ("tckts"/"urg"/"pprio") are normalized ("ntckts"/"nurg"/"npprior") before they are used to calculate job priorities ("prio").

Normalization maps each raw urgency/ticket/priority value into a range between 0 and 1 according to the used maximum and minimum values of the corresponding type. If the minimum and maximum value are (nearly) the same then the normalized value will be 0.5.

```
npprior = normalized(ppri)
nurg = normalized(urg)
ntckts = normalized(tckts)

prio = weight_priority * npprior +  weight_urgency * nurg +  weight_ticket * ntckts
```

The higher a job's priority value, the earlier it gets dispatched.

The urgency policy defines an urgency value for each job. The urgency value

```
urg = rrcontr + wtcontr + dlcontr
```

consists of the resource requirement contribution ("rrcontr"), the waiting time contribution ("wtcontr") and the deadline contribution ("dlcontr").

The resource requirement contribution is adding up all resource requirements of a job into a single numeric value.

```
rrcontr = Sum over all(hrr)
```

with an "hrr" for each hard resource request. Depending on the resource type two different methods are used to determine the value to be used for "hrr" here. For numeric type resource requests, the "hrr" represents how much of a resource a job requests (on a per-slot basis for pe jobs) and how "important" this resource is considered in comparison to other resources. This is expressed by the formula:

```
hrr = rurg * assumed_slot_allocation * request
```

where the resource's urgency value ("rurg") is as specified under **urgency** in sge_complex(5), the job's assumed_slot_allocation represents the number of slots supposedly assigned to the job, and the per-slot request is that which was specified using the -l qsub(1) option. For string type requests the formula is simply

```
hrr = "rurg"
```

and directly assigns the resource urgency value as specified under **urgency** in sge_complex(5).

The waiting time contribution represents a weighted weighting time of the jobs

```
wtcontr = waiting_time * weight_waiting_time
```

with the waiting time in seconds and the **weight_waiting_time** value as specified in *sge_sched_conf*(5).

The deadline contribution has an increasing effect as jobs approach their deadline initiation time (see the -dl option in qsub(1)). It is defined as the quotient of the **weight_deadline** value from *sge_sched_conf*(5) and the (steadily decreasing) free time in seconds until deadline initiation time

```
dlcontr = weight_deadline / free_time
```

or is set to 0 for non-deadline jobs. After the deadline passes, the value is static and equal to weight_deadline.

The ticket policy unites functional, override and share tree policies in the ticket value ("tckts"), as is defined as the sum of the specific ticket values ("ftckt"/"otckt"/"stckt") for each sub-policy (functional, override, share):

```
tckts = ftckt + otckt + stckt
```

The ticket policies provide a broad range of means for influencing both job dispatch and runtime priorities on a per job, per user, per project, and per department basis. See the Altair Grid Engine Installation and Administration Guide for details.

JOB RUN-TIME PRIORITY

The run-time priority can be dynamically adjusted in order to meet the goals set with the ticket policy. Dynamic run-time priority adjustment can be turned off (default) globally using **reprioritize_interval** in *sge_sched_conf*(5) If no dynamic run-time priority adjustment is done at a host level, the **priority** specification in *sge_queue_conf*(5) is in effect.

Note that urgency and POSIX priorities do **NOT** affect runtime priority.

SEE ALSO

sge_intro(1), *sge_complex*(5), *qstat*(1), *qsub*(1), *sge_sched_conf*(5), *sge_conf*(5) Altair Grid Engine Installation and Administration Guide

COPYRIGHT

sge_project

NAME

project - Altair Grid Engine project entry file format

DESCRIPTION

Jobs can be submitted to projects, and a project can be assigned with a certain level of importance via the functional or the override policy. This level of importance is then inherited by the jobs executing under that project.

A list of currently configured projects can be displayed via the qconf(1) **-sprjl** option. The contents of each enlisted project definition can be shown via the **-sprj** switch. The output follows the project format description. New projects can be created and existing can be modified via the **-aprj**, **-mprj** and **-dprj** options to qconf(1).

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

A project definition contains the following parameters:

name

The project name as defined for project_name in sge_types(1).

oticket

The amount of override tickets currently assigned to the project.

fshare

The current functional share of the project.

acl

A list of user access lists (ACLs - see sge_access_list(5)) referring to those users being allowed to submit jobs to the project.

If the **acl** parameter is set to NONE, all users are allowed to submit jobs to the project except for those listed in **xacl** parameter described below.

xacl

A list of user access lists (ACLs - see sge_access_list(5)) referring to those users being not allowed to submit jobs to the project.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), sge_access_list(5).

COPYRIGHT

sge_qstat

NAME

sge_qstat - Altair Grid Engine default qstat file format

DESCRIPTION

sge_qstat defines the command line switches that will be used by qstat by default. If available, the default sge_qstat file is read and processed by qstat(1).

There is a cluster global and a user private sge_qstat file. The user private file has the highest precedence and is followed by the cluster global sge_qstat file. Command line switches used with qstat(1) override all switches contained in the user private or cluster global sge_qstat file.

The format of the default files is:

- The default sge_qstat file may contain an arbitrary number of lines. Blank lines and lines with a '#' sign at the first column are skipped. Each line not to be skipped may contain any qstat(1) option as described in the Altair Grid Engine Reference Manual. More than one option per line is allowed.

EXAMPLES

The following is a simple example of a default sge_qstat file:

```
=====================================================
# Just show me my own running and suspended jobs
-s rs -u $user
=====================================================
```

Having defined a default sge_qstat file like this and using qstat as follows:

```
qstat
```

has the same effect as if qstat was executed with:

```
qstat -s rs -u <current_user>
```

FILES

`<sge_root>/<cell>/common/sge_qstat - global defaults file`

`$HOME/.sge_qstat - user private defaults file`

SEE ALSO

sge_intro(1), qstat(1), Altair Grid Engine Installation and Administration Guide

COPYRIGHT

sge_queue_conf

NAME

queue_conf - Altair Grid Engine queue configuration file format

DESCRIPTION

This manual page describes the format of the template file for the cluster queue configuration. Via the **-aq** and **-mq** options of the qconf(1) command, you can add cluster queues and modify the configuration of any queue in the cluster. Any of these change operations can be rejected, as a result of a failed integrity verification.

The queue configuration parameters take as values strings, integer decimal numbers or boolean, time and memory specifiers (see time_specifier and memory_specifier in sge_types(1)) as well as comma separated lists.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

The following list of parameters specifies the queue configuration file content:

qname

The name of the cluster queue as defined for queue_name in sge_types(1). As template default “template” is used.

hostlist

A list of host identifiers as defined for host_identifier in sge_types(1). For each host Altair Grid Engine maintains a queue instance for running jobs on that particular host. Large amounts of hosts can easily be managed by using host groups rather than by single host names. As list separators white-spaces and “,” can be used. (template default: NONE).

If more than one host is specified it can be desirable to specify divergences with the further below parameter settings for certain hosts. These divergences can be expressed using the enhanced queue configuration specifier syntax. This syntax builds upon the regular parameter specifier syntax separately for each parameter:

"["*host_identifier*=<parameters_specifier_syntax>"]" ["["*host_identifier*=<parameters_specifier_syntax>"]"]

note, even in the enhanced queue configuration specifier syntax an entry without brackets denoting the default setting is required and used for all queue instances where no divergences are specified. Tuples with a host group host_identifier override the default setting. Tuples with a host name host_identifier override both the default and the host group setting.

Note that also with the enhanced queue configuration specifier syntax a default setting is always needed for each configuration attribute; otherwise the queue configuration gets rejected. Ambiguous queue configurations with more than one attribute setting for a particular host are rejected. Configurations containing override values for hosts not enlisted under 'hostname' are accepted but are indicated by **-sds** of qconf(1). The cluster queue should contain an unambiguous specification for each configuration attribute of each queue instance specified under hostname in the queue configuration. Ambiguous configurations with more than one attribute setting resulting from overlapping 'hostname' are accepted but are indicated by **-sds** of qconf(1). The cluster queue should contain an unambiguous specification for each configuration attribute of each queue instance specified under hostname in the queue configuration. Ambiguous configurations with more than one attribute setting resulting from overlapping host groups are indicated by **-explain c** of qstat(1) and cause the queue instance with ambiguous configurations to enter the c(onfiguration ambiguous) state.

seq_no

In conjunction with the hosts load situation at a time this parameter specifies this queue's position in the scheduling order within the suitable queues for a job to be dispatched under consideration of **weight_queue_seqno** (see sge_sched_conf(5)).

qstat(1) reports queue information in the order defined by the value of the **seq_no**. Set this parameter to a monotonically increasing sequence. (type number; template default: 0).

load_thresholds

load_thresholds is a list of load thresholds. Already if one of the thresholds is exceeded no further jobs will be scheduled to the queues and qmon(1) will signal an overload condition for this node. Arbitrary load values being defined in the "host" and "global" complexes (see sge_complex(5) for details) can be used.

The syntax is that of a comma separated list with each list element consisting of the complex_name (see sge_types(1)) of a load value, an equal sign and the threshold value being intended to trigger the overload situation (e.g. load_avg=1.75,users_logged_in=5).

Note: Load values as well as consumable resources may be scaled differently for different hosts if specified in the corresponding execution host definitions (refer to sge_host_conf(5) for more information). Load thresholds are compared against the scaled load and consumable values.

suspend_thresholds

A list of load thresholds with the same semantics as that of the **load_thresholds** parameter (see above) except that exceeding one of the denoted thresholds initiates suspension of one of multiple jobs in the queue. See the **nsuspend** parameter below for details on the number of jobs which are suspended. There is an important relationship between the **suspend_threshold** and the **scheduler_interval**. If you have for example a suspend threshold on the np_load_avg, and the load exceeds the threshold, this does not have immediate effect. Jobs continue running until the next scheduling run, where the scheduler detects the threshold has been exceeded and sends an order to qmaster to suspend the job. The same applies for unsuspending.

nsuspend

The number of jobs which are suspended/enabled per time interval if at least one of the load thresholds in the **suspend_thresholds** list is exceeded or if no **suspend_threshold** is violated anymore respectively. **Nsuspend** jobs are suspended in each time interval until no **suspend_thresholds** are exceeded anymore or all jobs in the queue are suspended. Jobs are enabled in the corresponding way if the **suspend_thresholds** are no longer exceeded. The time interval in which the suspensions of the jobs occur is defined in **suspend_interval** below.

suspend_interval

The time interval in which further **nsuspend** jobs are suspended if one of the **suspend_thresholds** (see above for both) is exceeded by the current load on the host on which the queue is located. The time interval is also used when enabling the jobs. The syntax is that of a time_specifier in sge_types(1).

priority

The **priority** parameter specifies the nice(2) value at which jobs in this queue will be run. The type is number and the default is zero (which means no nice value is set explicitly). Negative values (up to -20) correspond to a higher scheduling priority, positive values (up to +20) correspond to a lower scheduling priority.

Note, the value of priority has no effect, if Altair Grid Engine adjusts priorities dynamically to implement ticket-based entitlement policy goals. Dynamic priority adjustment is switched off by default due to sge_conf(5) **reprioritize** being set to false.

min_cpu_interval

The time between two automatic checkpoints in case of transparently checkpointing jobs. The maximum of the time requested by the user via *qsub*(1) and the time defined by the queue configuration is used as checkpoint interval. Since checkpoint files may be considerably large and thus writing them to the file system may become expensive, users and

administrators are advised to choose sufficiently large time intervals. **min_cpu_interval** is of type time and the default is 5 minutes (which usually is suitable for test purposes only). The syntax is that of a time_specifier in sge_types(1).

processors

A set of processors in case of a multiprocessor execution host can be defined to which the jobs executing in this queue are bound. The value type of this parameter is a range description like that of the **-pe** option of qsub(1) (e.g. 1-4,8,10) denoting the processor numbers for the processor group to be used. Obviously the interpretation of these values relies on operating system specifics and is thus performed inside sge_execd(8) running on the queue host. Therefore, the parsing of the parameter has to be provided by the execution daemon and the parameter is only passed through sge_qmaster(8) as a string.

Currently, support is only provided for multiprocessor machines running Solaris, SGI multiprocessor machines running IRIX 6.2 and Digital UNIX multiprocessor machines. In the case of Solaris the processor set must already exist, when this processors parameter is configured. So the processor set has to be created manually. In the case of Digital UNIX only one job per processor set is allowed to execute at the same time, i.e. **slots** (see above) should be set to 1 for this queue.

qtype

The type of queue. Currently batch, interactive or a combination in a comma separated list or NONE.

Jobs that need to be scheduled immediately (qsh, qlogin, qrsh and qsub with option -now yes) can ONLY be scheduled on interactive queues.

The formerly supported types parallel and checkpointing are not allowed anymore. A queue instance is implicitly of type parallel/checkpointing if there is a parallel environment or a checkpointing interface specified for this queue instance in **pe_list**/**ckpt_list**. Formerly possible settings e.g.

qtype PARALLEL

could be transferred into

qtype NONE
pe_list pe_name

(type string; default: batch interactive).

pe_list

The list of administrator-defined parallel environment (see sge_pe(5)) names to be associated with the queue. The default is NONE.

jc_list

The list of job classes to be associated with this queue. The queue will only accept jobs that were derived from referenced job classes. If the queue should also accept jobs that were not derived from job classes then the keyword NO_JC has to be specified. If ANY_JC is referenced then jobs from all jobs classes will be accepted. NONE means that the queue does not accept any jobs. Default is the combination of the keywords ANY_JC and NO_JC.

ckpt_list

The list of administrator-defined checkpointing interface names (see ckpt_name in sge_types(1)) to be associated with the queue. The default is NONE.

rerun

Defines a default behavior for jobs which are aborted by system crashes or manual "violent" (via kill(1)) shutdown of the complete Altair Grid Engine system (including the sge_shepherd(8) of the jobs and their process hierarchy) on the queue host. As soon as sge_execd(8) is restarted and detects that a job has been aborted for such reasons it can be restarted if the jobs are restartable. A job may not be restartable, for example, if it updates databases (first reads then writes to the same record of a database/file) because the abortion of the job may have left the database in an inconsistent state. If the owner of a job wants to overrule the default behavior for the jobs in the queue the **-r** option of qsub(1) can be used.

The type of this parameter is boolean, thus either TRUE or FALSE can be specified. The default is FALSE, i.e. do not restart jobs automatically.

rerun_limit

Defines the number of times a job can be rescheduled, before the action, specified with rerun_limit_action is executed. Type is a number, valid values are 0 to 9999999. If 0 is set, the first reschedule event (e.g. **qmod -rj**) will trigger the rerun_limit_action, before the job is rescheduled for the first time. For parallel jobs only the master-queue is relevant. If a job is rescheduled multiple times into different queues, always only the settings of the current queue is considered after the reschedule-event. The number of reschedules is counted per job, not per queue.

rerun_limit_action

What action should be taken if a job reaches its rerun_limit.

Possible options are:

- *NONE* - Deactivate rerun_limit

- *DELETE* - Delete the job after it got into pending state after the final reschedule event. **qacct -j** will contain an additional accounting record, documenting that the job was deleted by the rerun_limit.
- *ERROR* - Set the job into error state and leave pending. **qstat -j** shows the reason for the error state.

slots

The maximum number of concurrently executing jobs allowed in any queue instance defined by the queue. Type is number, valid values are 0 to 9999999.

tmpdir

The **tmpdir** parameter specifies the absolute path to the base of the temporary directory filesystem. When sge_execd(8) launches a job, it creates a uniquely-named directory in this filesystem for the purpose of holding scratch files during job execution. At job completion, this directory and its contents are removed automatically. The environment variables TMPDIR and TMP are set to the path of each jobs scratch directory (type string; default: /tmp). Beginning with version 8.3.1p8 of Altair Grid Engine it is allowed to specify multiple directories separated by comma character (','). If multiple directories are specified then also in those, a uniquely-named directory will be created and the path will be available in the environment. The environment variables are named TMPDIR1, TMPDIR2, ...

Optional Linux mount namespace feature extension

For Linux kernels beginning with 2.4.19 allow automatic bind mounts with the Linux mount namespace feature. The following syntax will be used:

```
tmpdir /basedir(lns=true#mflags=rprivate#mdir=/tmp)[,/basedir2(...), etc.]

lns={true|false}
mflags={private|shared|unbindable|rprivate|rshared|runbindable}
mdir=<existing mount point where to 'bind mount' basedir
```

The separator between the parenthesis is # and no whitespace in between should be used. There are default values for mflags and mdir: 'mflags=rprivate and mdir=/tmp'.

For this reason these values can be omitted, so the simplest way to use linux mount namespaces is the following line:

```
tmpdir /basedir(lns=true)
```

If the parentheses suffix is not used Linux mount namespaces are disabled. For non Linux architectures the parentheses suffix is silently ignored.

shell

If either *posix_compliant* or *script_from_stdin* is specified as the **shell_start_mode** parameter in sge_conf(5) the **shell** parameter specifies the executable path of the command interpreter (e.g. sh(1) or csh(1)) to be used to process the job scripts executed in the queue. The definition of **shell** can be overruled by the job owner via the qsub(1) **-S** option.

The type of the parameter is string. The default is /bin/sh.

shell_start_mode

This parameter defines the mechanisms which are used to actually invoke the job scripts on the execution hosts. The following values are recognized:

- *unix_behavior* If a user starts a job shell script under UNIX interactively by invoking it just with the script name the operating system's executable loader uses the information provided in a comment such as '#!/bin/csh' in the first line of the script to detect which command interpreter to start to interpret the script. This mechanism is used by Altair Grid Engine when starting jobs if unix_behavior is defined as **shell_start_mode**.
- *posix_compliant* POSIX does not consider first script line comments such a '#!/bin/csh' as being significant. The POSIX standard for batch queuing systems (P1003.2d) therefore requires a compliant queuing system to ignore such lines but to use user specified or configured default command interpreters instead. Thus, if **shell_start_mode** is set to posix_compliant Altair Grid Engine will either use the command interpreter indicated by the **-S** option of the qsub(1) command or the **shell** parameter of the queue to be used (see above).
- *script_from_stdin* Setting the **shell_start_mode** parameter either to posix_compliant or unix_behavior requires you to set the umask in use for sge_execd(8) such that every user has read access to the active_jobs directory in the spool directory of the corresponding execution daemon. In case you have **prolog** and **epilog** scripts configured, they also need to be readable by any user who may execute jobs.
  If this violates your site's security policies you may want to set **shell_start_mode** to *script_from_stdin*. This will force Altair Grid Engine to open the job script as well as the epilogue and prologue scripts for reading into STDIN as root (if sge_execd(8) was started as root) before changing to the job owner's user account. The script is then fed into the STDIN stream of the command interpreter indicated by the **-S** option of the qsub(1) command or the **shell** parameter of the queue to be used (see above).
  Thus setting **shell_start_mode** to script_from_stdin also implies posix_compliant behavior. **Note**, however, that feeding scripts into the STDIN stream of a command interpreter may cause trouble if commands like rsh(1) are invoked inside a job script as they also process the STDIN stream of the command interpreter. These problems can usually be resolved by redirecting the STDIN channel of those commands to come from /dev/null (e.g. rsh host date < /dev/null). **Note also**, that any command-line options associated with the job are passed to the executing shell. The shell will only forward them to the job if they are not recognized as valid shell options.

The default for **shell_start_mode** is unix_behavior. Note, though, that the **shell_start_mode** can only be used for batch jobs submitted by qsub(1) and can't be used for interactive jobs submitted by qrsh(1), qsh(1), qlogin(1).

prolog

The executable path of a shell script that is started before execution of Altair Grid Engine jobs with the same environment setting as that for the Altair Grid Engine jobs to be started afterwards. An optional prefix "user@" specifies the user under which this procedure is to be started. The procedures standard output and the error output stream are written to the same file used also for the standard output and error output of each job. This procedure is intended as a means for the Altair Grid Engine administrator to automate the execution of general site specific tasks like the preparation of temporary file systems with the need for the same context information as the job. This queue configuration entry overwrites cluster global or execution host specific **prolog** definitions (see sge_conf(5)).

The default for **prolog** is the special value NONE, which prevents from execution of a prologue script. The special variables for constituting a command line are the same like in **prolog** definitions of the cluster configuration (see sge_conf(5)).

Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration.

Exit codes for the prolog attribute can be interpreted based on the following exit values:

```
0: Success
99: Reschedule job
100: Put job in error state
Anything else: Put queue in error state
```

epilog

The executable path of a shell script that is started after execution of Altair Grid Engine jobs with the same environment setting as that for the Altair Grid Engine jobs that has just completed. An optional prefix "user@" specifies the user under which this procedure is to be started. The procedures standard output and the error output stream are written to the same file used also for the standard output and error output of each job. This procedure is intended as a means for the Altair Grid Engine administrator to automate the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job. This queue configuration entry overwrites cluster global or execution host specific **epilog** definitions (see sge_conf(5)).

The default for **epilog** is the special value NONE, which prevents from execution of a epilogue script. The special variables for constituting a command line are the same like in

Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration. **prolog** definitions of the cluster configuration (see sge_conf(5)).

Exit codes for the epilog attribute can be interpreted based on the following exit values:

```
0: Success
99: Reschedule job
100: Put job in error state
Anything else: Put queue in error state
```

starter_method

The specified executable path will be used as a job starter facility responsible for starting batch jobs. The executable path will be executed instead of the configured shell to start the job. The job arguments will be passed as arguments to the job starter. The following environment variables are used to pass information to the job starter concerning the shell environment which was configured or requested to start the job.

- *SGE_STARTER_SHELL_PATH* The name of the requested shell to start the job
- *SGE_STARTER_SHELL_START_MODE* The configured **shell_start_mode**
- *SGE_STARTER_USE_LOGIN_SHELL* Set to "true" if the shell is supposed to be used as a login shell (see **login_shells** in sge_conf(5))

The starter_method will not be invoked for qsh, qlogin or qrsh acting as rlogin.

suspend_method

resume_method

terminate_method

These parameters can be used for overwriting the default method used by Altair Grid Engine for suspension, release of a suspension and for termination of a job. Per default, the signals SIGSTOP, SIGCONT and SIGKILL are delivered to the job to perform these actions. However, for some applications this is not appropriate.

If no executable path is given, Altair Grid Engine takes the specified parameter entries as the signal to be delivered instead of the default signal. A signal must be either a positive number or a signal name with **"SIG"** as prefix and the signal name as printed by kill -l (e.g. SIGTERM).

If an executable path is given (it must be an *absolute path* starting with a "/") then this command together with its arguments is started by Altair Grid Engine to perform the appropriate action. The following special variables are expanded at runtime and can be used (besides any other strings which have to be interpreted by the procedures) to constitute a command line:

Value	Description
$host	The name of the host on which the procedure is started.
$job_owner	The user name of the job owner.
$job_id	The Altair Grid Engine unique job identification number.
$job_name	The name of the job.

Value	Description
$queue	The name of the queue.
$job_pid	The pid of the job.

notify

The time waited between delivery of SIGUSR1/SIGUSR2 notification signals and suspend/kill signals if job was submitted with the qsub(1) -notify option. If this time is 00:00:00, no notification signal is sent.

owner_list

The **owner_list** enlists comma separated the login(1) user names (see user_name in sge_types(1)) of those users who are authorized to disable and suspend this queue through qmod(1) (Altair Grid Engine operators and managers can do this by default). It is customary to set this field for queues on interactive workstations where the computing resources are shared between interactive sessions and Altair Grid Engine jobs, allowing the workstation owner to have priority access. (default: NONE).

user_lists

The **user_lists** parameter contains a comma separated list of Altair Grid Engine user access list names as described in *sge_access_list*(5). Each user contained in at least one of the enlisted access lists has access to the queue. If the **user_lists** parameter is set to NONE (the default) any user has access being not explicitly excluded via the **xuser_lists** parameter described below. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the queue.

xuser_lists

The **xuser_lists** parameter contains a comma separated list of Altair Grid Engine user access list names as described in sge_access_list(5). Each user contained in at least one of the enlisted access lists is not allowed to access the queue. If the **xuser_lists** parameter is set to NONE (the default) any user has access. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the queue.

projects

The **projects** parameter contains a comma separated list of Altair Grid Engine projects (see sge_project(5)) that have access to the queue. Any project not in this list are denied access to the queue. If set to NONE (the default), any project has access that is not specifically excluded via the **xprojects** parameter described below. If a project is in both the **projects** and **xprojects** parameters, the project is denied access to the queue.

xprojects

The **xprojects** parameter contains a comma separated list of Altair Grid Engine projects (see sge_project(5)) that are denied access to the queue. If set to NONE (the default), no projects are denied access other than those denied access based on the **projects** parameter described above. If a project is in both the **projects** and **xprojects** parameters, the project is denied access to the queue.

subordinate_list

There are two different types of subordination:

- *Queuewise subordination* A list of Altair Grid Engine queue names as defined for queue_name in sge_types(1). Subordinate relationships are in effect only between queue instances residing at the same host. The relationship does not apply and is ignored when jobs are running in queue instances on other hosts. Queue instances residing on the same host will be suspended when a specified count of jobs is running in this queue instance. The list specification is the same as that of the **load_thresholds** parameter above, e.g. low_pri_q=5,small_q. The numbers denote the job slots of the queue that have to be filled in the superordinated queue to trigger the suspension of the subordinated queue. If no value is assigned a suspension is triggered if all slots of the queue are filled.

On nodes which host more than one queue, you might wish to accord better service to certain classes of jobs (e.g., queues that are dedicated to parallel processing might need priority over low priority production queues; default: NONE).

- *Slotwise subordination* The slotwise subordination provides a means to ensure that high priority jobs get the resources they need, while at the same time low priority jobs on the same host are not unnecessarily suspended, maximizing the host utilization. The slotwise subordination is designed to provide different preemption actions, but with the current implementation only suspension is provided. This means there is a subordination relationship defined between queues similar to the queuewise subordination, but if the suspend threshold is exceeded, not the whole subordinated queue is suspended, there are only single tasks running in single slots suspended.

Like with queuewise subordination, the subordination relationships are in effect only between queue instances residing at the same host. The relationship does not apply and is ignored when jobs and tasks are running in queue instances on other hosts.

The syntax is:

```
slots=<threshold>(<queue_list>)
```

where

```
<threshold> =a positive integer number
<queue_list>=<queue_def>[,<queue_list>]
<queue_def> =<queue>[:<seq_no>][:<action>]
<queue> =a Altair Grid Engine queue name as defined for queue_name in sge_types(1).
<seq_no> =sequence number among all subordinated queues of the same depth in the tree. The higher
sequence number, the lower is the priority of  the queue. Default is 0, which is the highest prio
<action> =the action to be taken if the threshold is exceeded. Supported is:
"sr": Suspend the task with the shortest run  time.
"lr": Suspend the task with the longest run  time.
Default is "sr".
```

Some examples of possible configurations and their functionalities:

a) The simplest configuration

subordinate_list slots=2(B.q)

which means the queue “B.q” is subordinated to the current queue (let’s call it “A.q”), the suspend threshold for all tasks running in “A.q” and “B.q” on the current host is two, the sequence number of “B.q” is “0” and the action is “suspend task with shortest run time first”. This subordination relationship looks like this:

A.q
|
B.q

This could be a typical configuration for a host with a dual core CPU. This subordination configuration ensures that tasks that are scheduled to “A.q” always get a CPU core for themselves, while jobs in “B.q” are not suspended as long as there are no jobs running in “A.q”.

If there is no task running in “A.q”, two tasks are running in “B.q” and a new task is scheduled to “A.q”, the sum of tasks running in “A.q” and “B.q” is three. Three is greater than two, this triggers the defined action. This causes the task with the shortest run time in the subordinated queue “B.q” to be suspended. After suspension, there is one task running in “A.q”, on task running in “B.q” and one task suspended in “B.q”.

b) A simple tree

subordinate_list slots=2(B.q:1, C.q:2)

This defines a small tree that looks like this:

```
A.q
|  |___
B.q   C.q
```

A use case for this configuration could be a host with a dual core CPU and queue “B.q” and “C.q” for jobs with different requirements, e.g. “B.q” for interactive jobs, “C.q” for batch jobs. Again, the tasks in “A.q” always get a CPU core, while tasks in “B.q” and “C.q” are suspended only if the threshold of running tasks is exceeded. Here the sequence number among the

queues of the same depth comes into play. Tasks scheduled to “B.q” can’t directly trigger the suspension of tasks in “C.q”, but if there is a task to be suspended, first “C.q” will be searched for a suitable task.

If there is one task running in “A.q”, one in “C.q” and a new task is scheduled to “B.q”, the threshold of “2” in “A.q”, “B.q” and “C.q” is exceeded. This triggers the suspension of one task in either “B.q” or “C.q”. The sequence number gives “B.q” a higher priority than “C.q”, therefore the task in “C.q” is suspended. After suspension, there is one task running in “A.q”, one task running in “B.q” and one task suspended in “C.q”.

c) More than two levels

Configuration of A.q: `subordinate\_list\ slots=2(B.q)`
Configuration of B.q: `subordinate\_list slots=2(C.q)`

looks like this:

A.q
|
B.q
|
C.q

These are three queues with high, medium and low priority. If a task is scheduled to “C.q”, first the subtree consisting of “B.q” and “C.q” is checked, the number of tasks running there is counted. If the threshold which is defined in “B.q” is exceeded, the job in “C.q” is suspended. Then the whole tree is checked, if the number of tasks running in “A.q”, “B.q” and “C.q” exceeds the threshold defined in “A.q” the task in “C.q” is suspended. This means, the effective threshold of any subtree is not higher than the threshold of the root node of the tree. If in this example a task is scheduled to “A.q”, immediately the number of tasks running in “A.q”, “B.q” and “C.q” is checked against the threshold defined in “A.q”.

d) Any tree

The computation of the tasks that are to be (un)suspended always starts at the queue instance that is modified, i.e. a task is scheduled to, a task ends at, the configuration is modified, a manual or other automatic (un)suspend is issued, except when it is a leaf node, like “D.q”, “E.q” and “G.q” in this example. Then the computation starts at its parent queue instance (like “B.q”, “C.q” or “F.q” in this example). From there first all running tasks in the whole subtree of this queue instance are counted. If the sum exceeds the threshold configured in the subordinate_list, in this subtree a task is searched to be suspended. Then the algorithm proceeds to the parent of this queue instance, counts all running tasks in the whole subtree below the parent and checks if the number exceeds the threshold configured at the parent’s subordinate_list. If so, it searches for a task to suspend in the whole subtree below the parent. And so on, until it did this computation for the root node of the tree.

complex_values

complex_values defines quotas for resource attributes managed via this queue. The syntax is the same as for **load_thresholds** (see above). The quotas are related to the resource

consumption of all jobs in a queue in the case of consumable resources (see sge_complex(5) for details on consumable resources) or they are interpreted on a per queue slot (see **slots** above) basis in the case of non-consumable resources. Consumable resource attributes are commonly used to manage free memory, free disk space or available floating software licenses while non-consumable attributes usually define distinctive characteristics like type of hardware installed.

For consumable resource attributes an available resource amount is determined by subtracting the current resource consumption of all running jobs in the queue from the quota in the **complex_values** list. Jobs can only be dispatched to a queue if no resource requests exceed any corresponding resource availability obtained by this scheme. The quota definition in the **complex_values** list is automatically replaced by the current load value reported for this attribute, if load is monitored for this resource and if the reported load value is more stringent than the quota. This effectively avoids oversubscription of resources.

Note: Load values replacing the quota specifications may have become more stringent because they have been scaled (see sge_host_conf(5)) and/or load adjusted (see sge_sched_conf(5)). The -F option of qstat(1) and the load display in the qmon(1) queue control dialog (activated by clicking on a queue icon while the "Shift" key is pressed) provide detailed information on the actual availability of consumable resources and on the origin of the values taken into account currently.

Note also: The resource consumption of running jobs (used for the availability calculation) as well as the resource requests of the jobs waiting to be dispatched either may be derived from explicit user requests during job submission (see the -l option to qsub(1)) or from a "default" value configured for an attribute by the administrator (see sge_complex(5)). The -r option to qstat(1) can be used for retrieving full detail on the actual resource requests of all jobs in the system.

For non-consumable resources Altair Grid Engine simply compares the job's attribute requests with the corresponding specification in **complex_values** taking the relation operator of the complex attribute definition into account (see sge_complex(5)). If the result of the comparison is "true", the queue is suitable for the job with respect to the particular attribute. For parallel jobs each queue slot to be occupied by a parallel task is meant to provide the same resource attribute value.

Note: Only numeric complex attributes can be defined as consumable resources and hence non-numeric attributes are always handled on a per queue slot basis.

host groups are indicated by **-explain c** of qstat(1) and cause the queue instance with ambiguous configurations to enter the c(onfiguration ambiguous) state.

seq_no

In conjunction with the hosts load situation at a time this parameter specifies this queue's position in the scheduling order within the suitable queues for a job to be dispatched under consideration of **weight_queue_seqno** (see sge_sched_conf(5)).

qstat(1) reports queue information in the order defined by the value of the **seq_no**. Set this parameter to a monotonically increasing sequence. (type number; template default: 0).

load_thresholds

load_thresholds is a list of load thresholds. Already if one of the thresholds is exceeded no further jobs will be scheduled to the queues and qmon(1) will signal an overload condition for this node. Arbitrary load values being defined in the “host” and “global” complexes (see sge_complex(5) for details) can be used.

The syntax is that of a comma separated list with each list element consisting of the complex_name (see sge_types(1)) of a load value, an equal sign and the threshold value being intended to trigger the overload situation (e.g. load_avg=1.75,users_logged_in=5).

Note: Load values as well as consumable resources may be scaled differently for different hosts if specified in the corresponding execution host definitions (refer to sge_host_conf(5) for more information). Load thresholds are compared against the scaled load and consumable values.

suspend_thresholds

A list of load thresholds with the same semantics as that of the **load_thresholds** parameter (see above) except that exceeding one of the denoted thresholds initiates suspension of one of multiple jobs in the queue. See the **nsuspend** parameter below for details on the number of jobs which are suspended. There is an important relationship between the **suspend_threshold** and the **scheduler_interval**. If you have for example a suspend threshold on the np_load_avg, and the load exceeds the threshold, this does not have immediate effect. Jobs continue running until the next scheduling run, where the scheduler detects the threshold has been exceeded and sends an order to qmaster to suspend the job. The same applies for unsuspending.

nsuspend

The number of jobs which are suspended/enabled per time interval if at least one of the load thresholds in the **suspend_thresholds** list is exceeded or if no **suspend_threshold** is violated anymore respectively. **Nsuspend** jobs are suspended in each time interval until no **suspend_thresholds** are exceeded anymore or all jobs in the queue are suspended. Jobs are enabled in the corresponding way if the **suspend_thresholds** are no longer exceeded. The time interval in which the suspensions of the jobs occur is defined in **suspend_interval** below.

suspend_interval

The time interval in which further **nsuspend** jobs are suspended if one of the **suspend_thresholds** (see above for both) is exceeded by the current load on the host on which the queue is located. The time interval is also used when enabling the jobs. The syntax is that of a time_specifier in sge_types(1).

priority

The **priority** parameter specifies the nice(2) value at which jobs in this queue will be run. The type is number and the default is zero (which means no nice value is set explicitly). Negative values (up to -20) correspond to a higher scheduling priority, positive values (up to +20) correspond to a lower scheduling priority.

Note, the value of priority has no effect, if Altair Grid Engine adjusts priorities dynamically to implement ticket-based entitlement policy goals. Dynamic priority adjustment is switched off by default due to sge_conf(5) **reprioritize** being set to false.

min_cpu_interval

The time between two automatic checkpoints in case of transparently checkpointing jobs. The maximum of the time requested by the user via *qsub*(1) and the time defined by the queue configuration is used as checkpoint interval. Since checkpoint files may be considerably large and thus writing them to the file system may become expensive, users and administrators are advised to choose sufficiently large time intervals. **min_cpu_interval** is of type time and the default is 5 minutes (which usually is suitable for test purposes only). The syntax is that of a time_specifier in sge_types(1).

processors

A set of processors in case of a multiprocessor execution host can be defined to which the jobs executing in this queue are bound. The value type of this parameter is a range description like that of the **-pe** option of qsub(1) (e.g. 1-4,8,10) denoting the processor numbers for the processor group to be used. Obviously the interpretation of these values relies on operating system specifics and is thus performed inside sge_execd(8) running on the queue host. Therefore, the parsing of the parameter has to be provided by the execution daemon and the parameter is only passed through sge_qmaster(8) as a string.

Currently, support is only provided for multiprocessor machines running Solaris, SGI multiprocessor machines running IRIX 6.2 and Digital UNIX multiprocessor machines. In the case of Solaris the processor set must already exist, when this processors parameter is configured. So the processor set has to be created manually. In the case of Digital UNIX only one job per processor set is allowed to execute at the same time, i.e. **slots** (see above) should be set to 1 for this queue.

qtype

The type of queue. Currently batch, interactive or a combination in a comma separated list or NONE.

Jobs that need to be scheduled immediately (qsh, qlogin, qrsh and qsub with option -now yes) can ONLY be scheduled on interactive queues.

The formerly supported types parallel and checkpointing are not allowed anymore. A queue instance is implicitly of type parallel/checkpointing if there is a parallel environment or a

checkpointing interface specified for this queue instance in **pe_list**/**ckpt_list**. Formerly possible settings e.g.

qtype PARALLEL

could be transferred into

qtype NONE
pe_list pe_name

(type string; default: batch interactive).

pe_list

The list of administrator-defined parallel environment (see sge_pe(5)) names to be associated with the queue. The default is NONE.

jc_list

The list of job classes to be associated with this queue. The queue will only accept jobs that were derived from referenced job classes. If the queue should also accept jobs that were not derived from job classes then the keyword NO_JC has to be specified. If ANY_JC is referenced then jobs from all jobs classes will be accepted. NONE means that the queue does not accept any jobs. Default is the combination of the keywords ANY_JC and NO_JC.

ckpt_list

The list of administrator-defined checkpointing interface names (see ckpt_name in sge_types(1)) to be associated with the queue. The default is NONE.

rerun

Defines a default behavior for jobs which are aborted by system crashes or manual "violent" (via kill(1)) shutdown of the complete Altair Grid Engine system (including the sge_shepherd(8) of the jobs and their process hierarchy) on the queue host. As soon as sge_execd(8) is restarted and detects that a job has been aborted for such reasons it can be restarted if the jobs are restartable. A job may not be restartable, for example, if it updates databases (first reads then writes to the same record of a database/file) because the abortion of the job may have left the database in an inconsistent state. If the owner of a job wants to overrule the default behavior for the jobs in the queue the **-r** option of qsub(1) can be used.

The type of this parameter is boolean, thus either TRUE or FALSE can be specified. The default is FALSE, i.e. do not restart jobs automatically.

rerun_limit

Defines the number of times a job can be rescheduled, before the action, specified with rerun_limit_action is executed. Type is a number, valid values are 0 to 9999999. If 0 is set, the first reschedule event (e.g. **qmod -rj**) will trigger the rerun_limit_action, before the job is rescheduled for the first time. For parallel jobs only the master-queue is relevant. If a job is rescheduled multiple times into different queues, always only the settings of the current queue is considered after the reschedule-event. The number of reschedules is counted per job, not per queue.

rerun_limit_action

What action should be taken if a job reaches its rerun_limit.

Possible options are:

- *NONE* - Deactivate rerun_limit
- *DELETE* - Delete the job after it got into pending state after the final reschedule event. **qacct -j** will contain an additional accounting record, documenting that the job was deleted by the rerun_limit.
- *ERROR* - Set the job into error state and leave pending. **qstat -j** shows the reason for the error state.

slots

The maximum number of concurrently executing jobs allowed in any queue instance defined by the queue. Type is number, valid values are 0 to 9999999.

tmpdir

The **tmpdir** parameter specifies the absolute path to the base of the temporary directory filesystem. When sge_execd(8) launches a job, it creates a uniquely-named directory in this filesystem for the purpose of holding scratch files during job execution. At job completion, this directory and its contents are removed automatically. The environment variables TMPDIR and TMP are set to the path of each jobs scratch directory (type string; default: /tmp). Beginning with version 8.3.1p8 of Altair Grid Engine it is allowed to specify multiple directories separated by comma character (','). If multiple directories are specified then also in those, a uniquely-named directory will be created and the path will be available in the environment. The environment variables are named TMPDIR1, TMPDIR2, ...

shell

If either *posix_compliant* or *script_from_stdin* is specified as the **shell_start_mode** parameter in sge_conf(5) the **shell** parameter specifies the executable path of the command interpreter (e.g. sh(1) or csh(1)) to be used to process the job scripts executed in the queue. The definition of **shell** can be overruled by the job owner via the qsub(1) **-S** option.

The type of the parameter is string. The default is /bin/sh.

shell_start_mode

This parameter defines the mechanisms which are used to actually invoke the job scripts on the execution hosts. The following values are recognized:

- *unix_behavior* If a user starts a job shell script under UNIX interactively by invoking it just with the script name the operating system's executable loader uses the information provided in a comment such as '#!/bin/csh' in the first line of the script to detect which command interpreter to start to interpret the script. This mechanism is used by Altair Grid Engine when starting jobs if unix_behavior is defined as **shell_start_mode**.
- *posix_compliant* POSIX does not consider first script line comments such a '#!/bin/csh' as being significant. The POSIX standard for batch queuing systems (P1003.2d) therefore requires a compliant queuing system to ignore such lines but to use user specified or configured default command interpreters instead. Thus, if **shell_start_mode** is set to posix_compliant Altair Grid Engine will either use the command interpreter indicated by the **-S** option of the qsub(1) command or the **shell** parameter of the queue to be used (see above).
- *script_from_stdin* Setting the **shell_start_mode** parameter either to posix_compliant or unix_behavior requires you to set the umask in use for sge_execd(8) such that every user has read access to the active_jobs directory in the spool directory of the corresponding execution daemon. In case you have **prolog** and **epilog** scripts configured, they also need to be readable by any user who may execute jobs.
  If this violates your site's security policies you may want to set **shell_start_mode** to *script_from_stdin*. This will force Altair Grid Engine to open the job script as well as the epilogue and prologue scripts for reading into STDIN as root (if sge_execd(8) was started as root) before changing to the job owner's user account. The script is then fed into the STDIN stream of the command interpreter indicated by the **-S** option of the qsub(1) command or the **shell** parameter of the queue to be used (see above).
  Thus setting **shell_start_mode** to script_from_stdin also implies posix_compliant behavior. **Note**, however, that feeding scripts into the STDIN stream of a command interpreter may cause trouble if commands like rsh(1) are invoked inside a job script as they also process the STDIN stream of the command interpreter. These problems can usually be resolved by redirecting the STDIN channel of those commands to come from /dev/null (e.g. rsh host date < /dev/null). **Note also**, that any command-line options associated with the job are passed to the executing shell. The shell will only forward them to the job if they are not recognized as valid shell options.

The default for **shell_start_mode** is unix_behavior. Note, though, that the **shell_start_mode** can only be used for batch jobs submitted by qsub(1) and can't be used for interactive jobs submitted by qrsh(1), qsh(1), qlogin(1).

prolog

The executable path of a shell script that is started before execution of Altair Grid Engine jobs with the same environment setting as that for the Altair Grid Engine jobs to be started

afterwards. An optional prefix "user@" specifies the user under which this procedure is to be started. The procedures standard output and the error output stream are written to the same file used also for the standard output and error output of each job. This procedure is intended as a means for the Altair Grid Engine administrator to automate the execution of general site specific tasks like the preparation of temporary file systems with the need for the same context information as the job. This queue configuration entry overwrites cluster global or execution host specific **prolog** definitions (see sge_conf(5)).

The default for **prolog** is the special value NONE, which prevents from execution of a prologue script. The special variables for constituting a command line are the same like in **prolog** definitions of the cluster configuration (see sge_conf(5)).

Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration.

Exit codes for the prolog attribute can be interpreted based on the following exit values:

```
0: Success
99: Reschedule job
100: Put job in error state
Anything else: Put queue in error state
```

epilog

The executable path of a shell script that is started after execution of Altair Grid Engine jobs with the same environment setting as that for the Altair Grid Engine jobs that has just completed. An optional prefix "user@" specifies the user under which this procedure is to be started. The procedures standard output and the error output stream are written to the same file used also for the standard output and error output of each job. This procedure is intended as a means for the Altair Grid Engine administrator to automate the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job. This queue configuration entry overwrites cluster global or execution host specific **epilog** definitions (see sge_conf(5)).

The default for **epilog** is the special value NONE, which prevents from execution of a epilogue script. The special variables for constituting a command line are the same like in

Scripts where the execution duration would exceed 2 minutes will be terminated. This timeout can be adjusted by defining **SCRIPT_TIMEOUT** as **execd_param** in the configuration. **prolog** definitions of the cluster configuration (see sge_conf(5)).

Exit codes for the epilog attribute can be interpreted based on the following exit values:

```
0: Success
99: Reschedule job
100: Put job in error state
Anything else: Put queue in error state
```

starter_method

The specified executable path will be used as a job starter facility responsible for starting batch jobs. The executable path will be executed instead of the configured shell to start the job. The job arguments will be passed as arguments to the job starter. The following environment variables are used to pass information to the job starter concerning the shell environment which was configured or requested to start the job.

- *SGE_STARTER_SHELL_PATH* The name of the requested shell to start the job
- *SGE_STARTER_SHELL_START_MODE* The configured **shell_start_mode**
- *SGE_STARTER_USE_LOGIN_SHELL* Set to "true" if the shell is supposed to be used as a login shell (see **login_shells** in sge_conf(5))

The starter_method will not be invoked for qsh, qlogin or qrsh acting as rlogin.

suspend_method

resume_method

terminate_method

These parameters can be used for overwriting the default method used by Altair Grid Engine for suspension, release of a suspension and for termination of a job. Per default, the signals SIGSTOP, SIGCONT and SIGKILL are delivered to the job to perform these actions. However, for some applications this is not appropriate.

If no executable path is given, Altair Grid Engine takes the specified parameter entries as the signal to be delivered instead of the default signal. A signal must be either a positive number or a signal name with **"SIG"** as prefix and the signal name as printed by kill -l (e.g. SIGTERM).

If an executable path is given (it must be an *absolute path* starting with a "/") then this command together with its arguments is started by Altair Grid Engine to perform the appropriate action. The following special variables are expanded at runtime and can be used (besides any other strings which have to be interpreted by the procedures) to constitute a command line:

Value	Description
$host	The name of the host on which the procedure is started.
$job_owner	The user name of the job owner.
$job_id	The Altair Grid Engine unique job identification number.
$job_name	The name of the job.
$queue	The name of the queue.
$job_pid	The pid of the job.

notify

The time waited between delivery of SIGUSR1/SIGUSR2 notification signals and suspend/kill signals if job was submitted with the qsub(1) -notify option. If this time is 00:00:00, no notification signal is sent.

owner_list

The **owner_list** enlists comma separated the login(1) user names (see user_name in sge_types(1)) of those users who are authorized to disable and suspend this queue through qmod(1) (Altair Grid Engine operators and managers can do this by default). It is customary to set this field for queues on interactive workstations where the computing resources are shared between interactive sessions and Altair Grid Engine jobs, allowing the workstation owner to have priority access. (default: NONE).

user_lists

The **user_lists** parameter contains a comma separated list of Altair Grid Engine user access list names as described in *sge_access_list*(5). Each user contained in at least one of the enlisted access lists has access to the queue. If the **user_lists** parameter is set to NONE (the default) any user has access being not explicitly excluded via the **xuser_lists** parameter described below. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the queue.

xuser_lists

The **xuser_lists** parameter contains a comma separated list of Altair Grid Engine user access list names as described in sge_access_list(5). Each user contained in at least one of the enlisted access lists is not allowed to access the queue. If the **xuser_lists** parameter is set to NONE (the default) any user has access. If a user is contained both in an access list enlisted in **xuser_lists** and **user_lists** the user is denied access to the queue.

projects

The **projects** parameter contains a comma separated list of Altair Grid Engine projects (see sge_project(5)) that have access to the queue. Any project not in this list are denied access to the queue. If set to NONE (the default), any project has access that is not specifically excluded via the **xprojects** parameter described below. If a project is in both the **projects** and **xprojects** parameters, the project is denied access to the queue.

xprojects

The **xprojects** parameter contains a comma separated list of Altair Grid Engine projects (see sge_project(5)) that are denied access to the queue. If set to NONE (the default), no projects are denied access other than those denied access based on the **projects** parameter

described above. If a project is in both the **projects** and **xprojects** parameters, the project is denied access to the queue.

subordinate_list

There are two different types of subordination:

- *Queuewise subordination* A list of Altair Grid Engine queue names as defined for queue_name in sge_types(1). Subordinate relationships are in effect only between queue instances residing at the same host. The relationship does not apply and is ignored when jobs are running in queue instances on other hosts. Queue instances residing on the same host will be suspended when a specified count of jobs is running in this queue instance. The list specification is the same as that of the **load_thresholds** parameter above, e.g. low_pri_q=5,small_q. The numbers denote the job slots of the queue that have to be filled in the superordinated queue to trigger the suspension of the subordinated queue. If no value is assigned a suspension is triggered if all slots of the queue are filled.

On nodes which host more than one queue, you might wish to accord better service to certain classes of jobs (e.g., queues that are dedicated to parallel processing might need priority over low priority production queues; default: NONE).

- *Slotwise subordination* The slotwise subordination provides a means to ensure that high priority jobs get the resources they need, while at the same time low priority jobs on the same host are not unnecessarily suspended, maximizing the host utilization. The slotwise subordination is designed to provide different preemption actions, but with the current implementation only suspension is provided. This means there is a subordination relationship defined between queues similar to the queuewise subordination, but if the suspend threshold is exceeded, not the whole subordinated queue is suspended, there are only single tasks running in single slots suspended.

Like with queuewise subordination, the subordination relationships are in effect only between queue instances residing at the same host. The relationship does not apply and is ignored when jobs and tasks are running in queue instances on other hosts.

The syntax is:

```
slots=<threshold>(<queue_list>)
```

where

```
<threshold> =a positive integer number
<queue_list>=<queue_def>[,<queue_list>]
<queue_def> =<queue>[:<seq_no>][:<action>]
<queue> =a Altair Grid Engine queue name as defined for queue_name in sge_types(1).
<seq_no> =sequence number among all subordinated queues of the same depth in the tree. The higher
sequence number, the lower is the priority of  the queue. Default is 0, which is the highest prio
```

```
<action> =the action to be taken if the threshold is exceeded. Supported is:
"sr": Suspend the task with the shortest run  time.
"lr": Suspend the task with the longest run  time.
Default is "sr".
```

Some examples of possible configurations and their functionalities:

a) The simplest configuration
   subordinate_list slots=2(B.q)

which means the queue “B.q” is subordinated to the current queue (let’s call it “A.q”), the suspend threshold for all tasks running in “A.q” and “B.q” on the current host is two, the sequence number of “B.q” is “0” and the action is “suspend task with shortest run time first”. This subordination relationship looks like this:

A.q
|
B.q

This could be a typical configuration for a host with a dual core CPU. This subordination configuration ensures that tasks that are scheduled to “A.q” always get a CPU core for themselves, while jobs in “B.q” are not suspended as long as there are no jobs running in “A.q”.

If there is no task running in “A.q”, two tasks are running in “B.q” and a new task is scheduled to “A.q”, the sum of tasks running in “A.q” and “B.q” is three. Three is greater than two, this triggers the defined action. This causes the task with the shortest run time in the subordinated queue “B.q” to be suspended. After suspension, there is one task running in “A.q”, on task running in “B.q” and one task suspended in “B.q”.

b) A simple tree
   subordinate_list slots=2(B.q:1, C.q:2)

This defines a small tree that looks like this:

```
A.q
| |____
B.q   C.q
```

A use case for this configuration could be a host with a dual core CPU and queue “B.q” and “C.q” for jobs with different requirements, e.g. “B.q” for interactive jobs, “C.q” for batch jobs. Again, the tasks in “A.q” always get a CPU core, while tasks in “B.q” and “C.q” are suspended only if the threshold of running tasks is exceeded. Here the sequence number among the queues of the same depth comes into play. Tasks scheduled to “B.q” can’t directly trigger the suspension of tasks in “C.q”, but if there is a task to be suspended, first “C.q” will be searched for a suitable task.

If there is one task running in “A.q”, one in “C.q” and a new task is scheduled to “B.q”, the threshold of “2” in “A.q”, “B.q” and “C.q” is exceeded. This triggers the suspension of one task in either “B.q” or “C.q”. The sequence number gives “B.q” a higher priority than “C.q”, therefore the task in “C.q” is suspended. After suspension, there is one task running in “A.q”, one task running in “B.q” and one task suspended in “C.q”.

c) More than two levels

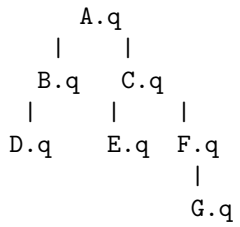
Configuration of A.q: `subordinate\_list slots=2(B.q)` Configuration of B.q: `subordinate\_list slots=2(C.q)`

looks like this:

```
A.q
|
B.q
|
C.q
```

These are three queues with high, medium and low priority. If a task is scheduled to “C.q”, first the subtree consisting of “B.q” and “C.q” is checked, the number of tasks running there is counted. If the threshold which is defined in “B.q” is exceeded, the job in “C.q” is suspended. Then the whole tree is checked, if the number of tasks running in “A.q”, “B.q” and “C.q” exceeds the threshold defined in “A.q” the task in “C.q” is suspended. This means, the effective threshold of any subtree is not higher than the threshold of the root node of the tree. If in this example a task is scheduled to “A.q”, immediately the number of tasks running in “A.q”, “B.q” and “C.q” is checked against the threshold defined in “A.q”.

d) Any tree



The computation of the tasks that are to be (un)suspended always starts at the queue instance that is modified, i.e. a task is scheduled to, a task ends at, the configuration is modified, a manual or other automatic (un)suspend is issued, except when it is a leaf node, like “D.q”, “E.q” and “G.q” in this example. Then the computation starts at its parent queue instance (like “B.q”, “C.q” or “F.q” in this example). From there first all running tasks in the whole subtree of this queue instance are counted. If the sum exceeds the threshold configured in the subordinate_list, in this subtree a task is searched to be suspended. Then the algorithm proceeds to the parent of this queue instance, counts all running tasks in the whole subtree below the parent and checks if the number exceeds the threshold configured at the parent’s subordinate_list. If so, it searches for a task to suspend in the whole subtree below the parent. And so on, until it did this computation for the root node of the tree.

complex_values

complex_values defines quotas for resource attributes managed via this queue. The syntax is the same as for **load_thresholds** (see above). The quotas are related to the resource

consumption of all jobs in a queue in the case of consumable resources (see sge_complex(5) for details on consumable resources) or they are interpreted on a per queue slot (see **slots** above) basis in the case of non-consumable resources. Consumable resource attributes are commonly used to manage free memory, free disk space or available floating software licenses while non-consumable attributes usually define distinctive characteristics like type of hardware installed.

For consumable resource attributes an available resource amount is determined by subtracting the current resource consumption of all running jobs in the queue from the quota in the **complex_values** list. Jobs can only be dispatched to a queue if no resource requests exceed any corresponding resource availability obtained by this scheme. The quota definition in the **complex_values** list is automatically replaced by the current load value reported for this attribute, if load is monitored for this resource and if the reported load value is more stringent than the quota. This effectively avoids oversubscription of resources.

Note: Load values replacing the quota specifications may have become more stringent because they have been scaled (see sge_host_conf(5)) and/or load adjusted (see sge_sched_conf(5)). The -F option of qstat(1) and the load display in the qmon(1) queue control dialog (activated by clicking on a queue icon while the "Shift" key is pressed) provide detailed information on the actual availability of consumable resources and on the origin of the values taken into account currently.

Note also: The resource consumption of running jobs (used for the availability calculation) as well as the resource requests of the jobs waiting to be dispatched either may be derived from explicit user requests during job submission (see the -l option to qsub(1)) or from a "default" value configured for an attribute by the administrator (see sge_complex(5)). The -r option to qstat(1) can be used for retrieving full detail on the actual resource requests of all jobs in the system.

For non-consumable resources Altair Grid Engine simply compares the job's attribute requests with the corresponding specification in **complex_values** taking the relation operator of the complex attribute definition into account (see sge_complex(5)). If the result of the comparison is "true", the queue is suitable for the job with respect to the particular attribute. For parallel jobs each queue slot to be occupied by a parallel task is meant to provide the same resource attribute value.

Note: Only numeric complex attributes can be defined as consumable resources and hence non-numeric attributes are always handled on a per queue slot basis.

The default value for this parameter is NONE, i.e. no administrator defined resource attribute quotas are associated with the queue.

calendar

specifies the **calendar** to be valid for this queue or contains NONE (the default). A calendar defines the availability of a queue depending on time of day, week and year. Please refer to sge_calendar_conf(5) for details on the Altair Grid Engine calendar facility.

Note: Jobs can request queues with a certain calendar model via a "-l c=" option to qsub(1).

initial_state

defines an initial state for the queue either when adding the queue to the system for the first time, on start-up of the sge_execd(8) on the host on which the queue resides, sge_qmaster(8) restart or migration (due to a failure of the sge_qmaster(8) daemon or underlying host). Possible values are:

Value	Description
default	The queue is enabled when adding the queue or is reset to the previous status (this corresponds to the behavior in earlier Altair Grid Engine releases not supporting initial_state).
enabled	The queue is enabled in either case. This is equivalent to a manual and explicit 'qmod -e' command (see qmod(1)).
disabled	The queue is disable in either case. This is equivalent to a manual and explicit 'qmod -d' command (see qmod(1)).
execd_enabled	The queue is enabled only on start-up of the sge_execd(8), not on sge_qmaster(8) restart or migration.
execd_disabled	The queue is disabled only on start-up of the sge_execd(8), not on sge_qmaster(8) restart or migration.

RESOURCE LIMITS

The first three resource limit parameters, **s_rt**, **h_rt** and **d_rt**, are implemented by Altair Grid Engine. They define the "real time" or also called "elapsed" or "wall clock" time having passed since the start of the job. If **h_rt** is exceeded by a job running in the queue, it is aborted via the SIGKILL signal (see kill(1)). If **s_rt** is exceeded, the job is first "warned" via the SIGUSR1 signal (which can be caught by the job) and finally aborted after the notification time defined in the queue configuration parameter **notify** (see above) has passed. In cases when **s_rt** is used in combination with job notification it might be necessary to configure a signal other than SIGUSR1 using the NOTIFY_KILL and NOTIFY_SUSP execd_params (see sge_conf(5)) so that the jobs' signal-catching mechanism can "differ" the cases and react accordingly.

All three limits are used by the Altair Grid Engine scheduler for planning resource reservation and backfilling. The **d_rt** (dynamic runtime limit) is not enforced, exceeding **d_rt** has no effect on the job. It is only used for planning purposes in the Altair Grid Engine scheduler, use it when specifying a run time is required for resource reservation and backfilling but no precise estimate can be given. **d_rt** can be modified with qalter for running and pending jobs.

The resource limit parameters **s_cpu** and **h_cpu** are implemented by Altair Grid Engine as

a job limit. They impose a limit on the amount of combined CPU time consumed by all the processes in the job. If **h_cpu** is exceeded by a job running in the queue, it is aborted via a SIGKILL signal (see kill(1)). If **s_cpu** is exceeded, the job is sent a SIGXCPU signal which can be caught by the job. If you wish to allow a job to be "warned" so it can exit gracefully before it is killed then you should set the **s_cpu** limit to a lower value than **h_cpu**. For parallel processes, the limit is applied per slot which means that the limit is multiplied by the number of slots being used by the job before being applied.

For sequential jobs, the resource limit parameters **s_vmem** and **h_vmem** are implemented by Altair Grid Engine as a job limit. They impose a limit on the amount of combined virtual memory consumed by all the processes in the job. If **h_vmem** is exceeded by a job running in the queue, it is aborted via a SIGKILL signal (see kill(1).) If the **cgroups_params** parameter h_vmem_limit is set to true the specified h_vmem limit will only be handled by cgroups (see sge_conf(5).) If **s_vmem** is exceeded, the job is sent a SIGXCPU signal which can be caught by the job. If you wish to allow a job to be "warned" so it can exit gracefully before it is killed then you should set the **s_vmem** limit to a lower value than **h_vmem**. For parallel jobs, the limit is applied per slot which means that the limit is multiplied by the number of slots being used by the job before being applied.

The resource limit parameters **s_rss** and **h_rss** are implemented by Altair Grid Engine as a job limit. They impose a limit on the amount of resident memory consumed by all the processes in the job. If **h_rss** is exceeded by a job running in the queue, it is aborted via a SIGKILL signal (see kill(1).) If **s_rss** is exceeded, the job is sent a SIGXCPU signal which can be caught by the job. If you wish to allow a job to be "warned" so it can exit gracefully before it is killed then you should set the **s_rss** limit to a lower value than **h_rss**. For parallel processes, the limit is applied per slot which means that the limit is multiplied by the number of slots being used by the job before being applied.

The remaining parameters in the queue configuration template specify per job soft and hard resource limits as implemented by the setrlimit(2) system call. See this manual page on your system for more information.

The value type for the CPU time limits **s_cpu** and **h_cpu** is time_specifier, the value type for the other limits is memory_specifier as described in the sge_types(1) manual page. For each limit field, the keyword "INFINITY" (which means RLIM_INFINITY as described in the setrlimit(2) manual page) can be used to set the resource limit to "unlimited". "INFINITY" is the default for all limit fields.

Following limits will be set and should be supported by all systems:

Limit	Parameters	Description
RLIMIT_CPU	s_cpu, h_cpu	CPU time limit in seconds.
RLIMIT_FSIZE	s_fsize, h_fsize	File size limit in bytes.
RLIMIT_DATA	s_data, h_data	Maximum size of process data segment in bytes.
RLIMIT_STACK	s_stack, h_stack	Maximum size of process stack in bytes.
RLIMIT_CORE	s_core, h_core	Maximum size of process core file in bytes.

Note: Not all systems support the same limits. Please check the setrlimit(2) man page for your operating system. Following limits are set if the operating system supports it:

Limit	Parameters	Description
RLIMIT_NOFILE	s_descriptors, h_descriptors	Maximum file descriptors for the process.
RLIMIT_NPROC	s_max_proc, h_max_proc	Maximum number of processes/threads for the process.
RLIMIT_MEMLOCK	s_memorylocked, h_memorylocked	Maximum bytes locked into RAM of process.
RLIMIT_LOCKS	s_locks, h_locks	Maximum number of locks of process.
RLIMIT_RSS	s_rss, h_rss	Maximum resident set size in bytes.
RLIMIT_VMEM/AS	s_vmem, h_vmem	Maximum size of virtual memory in bytes (See note below).

Note: Hard and soft values for descriptors, maxproc, memorylocked and locks are set via **execd_params** in the configuration. Please have a look at the sge_conf(5) manual page for resource limits.

Note: Special handling for s_vmem and h_vmem (Maximum size of virtual memory): If the operating system supports the limit RLIMIT_VMEM the values from s_vmem and h_vmem are set into this limit. If RLIMIT_VMEM is **not** supported but the operating system supports the RLIMIT_AS the values from s_vmem and h_vmem are set into this limit.

SEE ALSO

sge_intro(1), sge_types(1), csh(1), qconf(1), qmon(1), qstat(1), qsub(1), sh(1), nice(2), setrlimit(2), sge_access_list(5), sge_calendar_conf(5), sge_conf(5), sge_complex(5), sge_host_conf(5), sge_sched_conf(5), sge_execd(8), sge_qmaster(8), sge_shepherd(8).

COPYRIGHT

sge_reporting

NAME

sge_reporting(5) - Altair Grid Engine reporting file format

DESCRIPTION

A Altair Grid Engine system writes a reporting file to $SGE_ROOT/default/common/reporting. The reporting file contains data that can be used for accounting, monitoring and analysis purposes. It contains information about the cluster (hosts, queues, load values, consumables, etc.), about the jobs running in the cluster and about sharetree configuration and usage. All information is time related, events are dumped to the reporting file in a configurable interval. It allows to monitor a “real time” status of the cluster as well as historical analysis.

FORMAT

The reporting file is an ASCII file. Each line contains one record, and the fields of a record are separated by a delimiter (:). The reporting file contains records of different type. Each record type has a specific record structure.

The first two fields contain the same meta data for all reporting records:

time: Time (64bit GMT UNIX timestamp in milliseconds) when the record was created.

record type: Type of the accounting record. The different types of records and their structure are described in the following text.

The reporting records themselves start at the 3rd field of a line in the reporting file. The **numbers in brackets** represent the position within each reporting record. E.g. (3) means the value is shown in the 3rd field of the reporting record, i.e. it is the 5th field of the line in the reporting file.

new_job

The *new_job* record is written whenever a new job enters the system (usually by a submitting command). It has the following fields:

submission_time (1): Time (64bit GMT UNIX time stamp in milliseconds) when the job was submitted.

job_number (2): The job number.

task_number (3): The array task ID. Always has the value -1 for *new_job* records (as we don't have array tasks yet).

pe_taskid (4): The task ID of parallel tasks. Always has the value *NONE* for *new_job* records.

job_name (5): The job name (from **-N** submission option)

owner (6): The job owner.

group (7): The UNIX group of the job owner.

project (8): The project the job is running in.

department (9): The department the job owner is in.

account (10): The account string specified for the job (from **-A** submission option).

priority (11): The job priority (from **-p** submission option).

job_class (12): If the job has been submitted into a job class, the name of the job class, otherwise "".

submit_host (13): The submit host name.

submit_cmd (14)

The command line used for job submission. As the delimiter used by the reporting file (colon ":") can be part of the command line all colons in the command line are replaced by ASCII code 255. When reading the reporting file characters with ASCII code 255 have to be converted back to colon. Line feeds being part of the command line will be replaced by a space character. For jobs submitted via the DRMAA interface or via qmon graphical user interface the reporting file contains *NONE* as submit_cmd.

hard_resources (15): The hard requests of the job (from **[-hard] -l** submission option).

soft_resources (16): The soft requests (from **-soft -l** submission option).

hard_queues (17): The hard requested queues (from **[-hard] -q** submission option).

soft_queues (18): The soft requested queues (from **-soft -q** submission option).

job_log

The job_log record is written whenever a job, an array task or a pe tasks is changing status. A status change can be the transition from pending to running, but can also be triggered by user actions like suspension of a job. It has the following fields:

event_time (1): Time (64bit GMT UNIX time stamp in milliseconds) when the event was generated.

event (2): A one word description of the event.

job_number (3): The job number.

task_number (4): The array task ID. Always has the value -1 for *new_job* records (as we don't have array tasks yet).

pe_taskid (5): The task ID of parallel tasks. Always has the value *NONE* for *new_job* records.

state (6): The state of the job after the event was processed.

user (7): The user who initiated the event (or special usernames “master”, “scheduler” and “execution daemon” for actions of the system itself like scheduling jobs, executing jobs etc.).

host (8): The host from which the action was initiated (e.g. the submit host, the qmaster host, etc.).

state_time (9): Reserved field for later use.

priority (10): The job priority (from **-p** submission option).

submission_time (11): Time (64bit GMT UNIX time stamp in milliseconds) when the job was submitted.

job_name (12): The job name (from **-N** submission option)

owner (13): The job owner.

group (14): The UNIX group of the job owner.

project (15): The project the job is running in.

department (16): The department the job owner is in.

account (17): The account string specified for the job (from **-A** submission option).

job_class (18): If the job has been submitted into a job class, the name of the job class, otherwise "".

message (19): A message describing the reported action.

online_usage

Online usage records are written per array task or pe task of running jobs if online usage reporting is configured in the global cluster configuration, see also sge_conf(5) or per job via the **-rou** option, see also submit(1). An online usage record contains the following fields:

report_time: Time (64bit GMT UNIX time stamp in milliseconds) when the usage values were generated by sge_execd.

job_number (1): The job number.

task_number (2): The array task ID.

pe_taskid (3): The task ID of parallel tasks.

usage (4): Comma separated list of name=value tuples.

acct

Records of type acct are accounting records. Normally, they are written whenever a job, a task of an array job, or the task of a parallel job terminates. However, for long running jobs an intermediate acct record is created once a day after a midnight. This results in multiple accounting records for a particular job and allows for a fine-grained resource usage monitoring over time. Accounting records comprise the following fields:

qname (1): Name of the cluster queue in which the job has run.

hostname (2): Name of the execution host.

group (3): The effective group ID of the job owner when executing the job.

owner (4): Owner of the Altair Grid Engine job.

job_name (5): Job name.

job_number (6): Job identifier - job number.

account (7): An account string as specified by the qsub(1) or qalter(1) **-A** option.

priority (8): Priority value assigned to the job corresponding to the priority parameter in the queue configuration (see sge_queue_conf(5)).

submission_time (9): Submission time (64bit GMT UNIX time stamp in milliseconds).

start_time (10): Start time (64bit GMT UNIX time stamp in milliseconds).

end_time (11): End time (64bit GMT UNIX time stamp in milliseconds).

failed (12): Indicates the problem which occurred in case a job could not be started on the execution host (e.g. because the owner of the job did not have a valid account on that machine). If Altair Grid Engine tries to start a job multiple times, this may lead to multiple entries in the accounting file corresponding to the same job ID.

exit_status (13): Exit status of the job script (or Altair Grid Engine specific status in case of certain error conditions).

ru_wallclock (14): Difference between end_time and start_time (see above).

The remainder of the accounting entries follows the contents of the standard UNIX rusage structure as described in getrusage(2). Depending on the operating system where the job was executed some of the fields may be 0. The following entries are provided:

- **ru_utime** (15)
- **ru_stime** (16)
- **ru_maxrss** (17)
- **ru_ixrss** (18)
- **ru_ismrss** (19)
- **ru_idrss** (20)
- **ru_isrss** (21)
- **ru_minflt** (22)
- **ru_majflt** (23)
- **ru_nswap** (24)
- **ru_inblock** (25)
- **ru_oublock** (26)
- **ru_msgsnd** (27)
- **ru_msgrcv** (28)
- **ru_nsignals** (29)
- **ru_nvcsw** (30)
- **ru_nivcsw** (31)

On Windows, only the values *ru_wallclock*, *ru_utime* and *ru_stime* are accounted. These values are the final usage values of the Windows Job object that is used to reflect the Altair Grid Engine job, not the sum of the usage of all processes.

project (32): The project which was assigned to the job.

department (33): The department which was assigned to the job.

granted_pe (34): The parallel environment which was selected for that job.

slots (35): The number of slots which were dispatched to the job by the scheduler.

task_number (36): Array job task index number.

cpu (37): The cpu time usage in seconds.

mem (38): The integral memory usage in Gbytes seconds.

io (39): The amount of data transferred in Gbytes. On Linux data transferred means all bytes read and written by the job through the read(), pread(), write() and pwrite() systems calls. On Windows this is the sum of all bytes transferred by the job by doing write, read and other operations. It's not documented what these other operations are.

category (40): A string specifying the job category.

iow (41): The io wait time in seconds.

ioops (54): The number of io operations.

pe_taskid (42): If this identifier is set the task was part of a parallel job and was passed to Altair Grid Engine via the qrsh **-inherit** interface.

maxvmem (43): The maximum vmem size in bytes.

arid (44): Advance reservation identifier. If the job used resources of an advance reservation then this field contains a positive integer identifier otherwise the value is "0" .

ar_submission_time (45): If the job used resources of an advance reservation then this field contains the submission time (64bit GMT UNIX time stamp in milliseconds) of the advance reservation, otherwise the value is "0" .

job_class (46): If the job has been running in a job class, the name of the job class, otherwise *NONE* .

qdel_info (47): If the job (the array task) has been deleted via qdel, `<username>@<hostname>`, else *NONE*. If qdel was called multiple times, every invocation is recorded in a comma separated list.

maxrss (48): The maximum resident set size in bytes.

maxpss (49): The maximum proportional set size in bytes.

submit_host (50): The submit host name.

cwd (51): The working directory the job ran in as specified with qsub/qalter switches **-cwd** and **-wd**. As the delimiter used by the accounting file (colon ":") can be part of the working directory all colons in the working directory are replaced by ASCII code 255.

submit_cmd (52): The command line used for job submission. As the delimiter used by the reporting file (colon ":") can be part of the command line all colons in the command line are replaced by ASCII code 255. When reading the reporting file characters with ASCII code 255 have to be converted back to colon. Line feeds being part of the command line will be replaced by a space character. For jobs submitted via the DRMAA interface or via qmon graphical user interface the reporting file contains *NONE* as *submit_cmd*.

wallclock (53): The wallclock time the job spent in running state. Time spent in prolog, epilog, pe_start and pe_stop scripts also counts as wallclock time. Times during which the job was suspended are not counted as wallclock time. This value is measured by execution daemon.

ioops (54): The number of io operations.

bound_cores (55): The socket and core number which was bound by this job (depends on ENABLE_BINDING_RECORDS qmaster_params setting).

resource_map (56): The assigned Resource Map ids (depends on ENABLE_BINDING_RECORDS qmaster_params setting).

devices (57): The assigned devices (from Resource Map **device** parameter, depends on ENABLE_BINDING_RECORDS qmaster_params setting).

gpus (58): The assigned GPUs (from Resource Map **amd_id**, **cuda_id** or **xpu_id** parameter).

gpu_usage (59): The GPU specific usage values that were reported for the job.

failcnt (60): Number of memory limit hits (cgroups).

memsw.failcnt (61): Number of memory and swap limit hits (cgroups).

max_usage_in_bytes (62): Maximum recorded memory usage (cgroups).

memsw.max_usage_in_bytes (63): Maximum recorded memory and swap usage (cgroups).

max_cgroups_memory (64): Maximum of the cgroups_memory load value. cgroups_memory is calculated from values in the cgroups file memory.stat as

cgroups_memory = total_active_anon + total_inactive_anon + total_unevictable

hold_jid (65): The job ids which this job holds for.

orig_exec_time (66): The initially requested execution time when the job was submitted (from **-a date_time** submission option).

exec_time (67): The requested execution time (from **-a date_time** submission option).

ar_name (68): The name of the Advance Reservation of the job.

hard_resources (69): The hard requests of the job (from **[-hard] -l** submission option).

soft_resources (70): The soft requests (from **-soft -l** submission option).

hard_queues (71): The hard requested queues (from **[-hard] -q** submission option).

soft_queues (72): The soft requested queues (from **-soft -q** submission option).

granted_req. (73): The granted requests (may be different to the initial requests of the job, depends on ENABLE_BINDING_RECORDS qmaster_params setting). Each “attribute=value” pair is prefixed by “array task ID,parallel task ID:”.

exec_host_list (74): The execution hosts the job was running on.

queue

Records of type queue contain state information for queues (queue instances). A queue record has the following fields:

qname (1): The cluster queue name.

hostname (2): The hostname of a specific queue instance.

report_time (3): The time (64bit GMT UNIX time stamp in milliseconds) when a state change was triggered.

state (4): The new queue state.

queue_consumable

A queue_consumable record contains information about queue consumable values in addition to queue state information:

qname (1): The cluster queue name.

hostname (2): The hostname of a specific queue instance.

report_time (3): The time (64bit GMT UNIX time stamp in milliseconds) when a state change was triggered.

state (4): The new queue state.

consumables (5): Description of consumable values. Information about multiple consumables is separated by space. A consumable description has the format `<name>=<actual_value>=<configured_value>`.

host

A host record contains information about hosts and host load values. It contains the following information:

hostname (1): The name of the host.

report_time (2): The time (64bit GMT UNIX time stamp in milliseconds) when the reported information was generated.

state (3): The new host state. Currently, Altair Grid Engine doesn't track a host state, the field is reserved for future use. Always contains the value X.

load values (4): Description of load values. Information about multiple load values is separated by space. A load value description has the format `<name>=<actual_value>`.

host_consumable

A *host_consumable* record contains information about hosts and host consumables. Host consumables can for example be licenses. It contains the following information:

hostname (1): The name of the host.

report_time (2): The time (64bit GMT UNIX time stamp in milliseconds) when the reported information was generated.

state (3): The new host state. Currently, Altair Grid Engine doesn't track a host state, the field is reserved for future use. Always contains the value X.

consumables (4): Description of consumable values. Information about multiple consumables is separated by space. A consumable description has the format `<name>=<actual_value>=<configured_value>`.

sharelog

The Altair Grid Engine qmaster can dump information about sharetree configuration and use to the reporting file. The parameter sharelog sets an interval in which sharetree information will be dumped. It is set in the format HH:MM:SS. A value of 00:00:00 configures qmaster not to dump sharetree information. Intervals of several minutes up to hours are sensible values for this parameter. The record contains the following fields

current_time (1): The present time

usage_time (2): The time used so far

node_name (3): The node name

user_name (4): The user name

project_name (5): The project name

shares (6): The total shares

job_count (7): The job count

level (8): The percentage of shares used

total (9): The adjusted percentage of shares used

long_target_share (10): The long target percentage of resource shares used

short_target_share (11): The short target percentage of resource shares used

actual_share (12): The actual percentage of resource shares used

usage (13): The combined shares used

cpu (14): The cpu used

mem (15): The memory used

io (16): The IO used

long_target_cpu (17): The long target cpu used

long_target_mem (18): The long target memory used

long_target_io (19): The long target IO used

new_ar

A *new_ar* record contains information about advance reservation objects. Entries of this type will be added if an advance reservation is created. It contains the following information:

submission_time: The time (64bit GMT UNIX time stamp in milliseconds) when the advance reservation was created.

ar_number (1): The advance reservation number identifying the reservation.

ar_owner (2): The owner of the advance reservation.

ar_attribute

The *ar_attribute* record is written whenever a new advance reservation was added or the attribute of an existing advance reservation has changed. It has following fields.

event_time (1): The time (64bit GMT UNIX time stamp in milliseconds) when the event was generated.

submission_time (2): The time (64bit GMT UNIX time stamp in milliseconds) when the advance reservation was created.

ar_number (3): The advance reservation number identifying the reservation.

ar_name (4): Name of the advance reservation.

ar_account (5): An account string which was specified during the creation of the advance reservation.

ar_start_time (6): Start time.

ar_end_time (7): End time.

ar_granted_pe (8): The parallel environment which was selected for an advance reservation.

ar_granted_resources (9): The granted resources which were selected for an advance reservation.

ar_sr_cal_week (10): In case of standing reservation the week calendar describing the reservation points, max. 2048 characters. See also the **-cal_week** option in the qrsub(1) man page.

ar_sr_depth (11): In case of standing reservation the SR depth (the number of reservations being done at a time). See also the **-cal_depth** option in the qrsub(1) man page.

ar_sr_jmp (12): In case of standing reservation the number of un-allocatable reservations to accept See also the **-cal_jmp** option in the qrsub(1) man page.

ar_log

The *ar_log* record is written whenever a advance reservation is changing status. A status change can be from pending to active, but can also be triggered by system events like host outage. It has following fields.

ar_state_change_time (1): The time (64bit GMT UNIX time stamp in milliseconds) when the event occurred which caused a state change.

submission_time (2): The time (64bit GMT UNIX time stamp in milliseconds) when the advance reservation was created.

ar_number (3): The advance reservation number identifying the reservation.

ar_state (4): The new state.

ar_event (5): An event ID identifying the event which caused the state change.

ar_message (6): A message describing the event which caused the state change.

ar_sr_id (7): In case of standing reservation the SR ID (a number >= 0), in case of advance reservation -1.

ar_acct

The *ar_acct* records are accounting records which are written for every queue instance whenever a advance reservation terminates. Advance reservation accounting records comprise following fields.

ar_termination_time (1): The time (64bit GMT UNIX time stamp in milliseconds) when the advance reservation terminated.

submission_time (2): The time (64bit GMT UNIX time stamp in milliseconds) when the advance reservation was created.

ar_number (3): The advance reservation number identifying the reservation.

ar_qname (4): Cluster queue name which the advance reservation reserved.

ar_hostname (5): The name of the execution host.

ar_slots (6): The number of slots which were reserved.

ar_sr_id (7): In case of standing reservation the SR ID (a number >= 0), in case of advance reservation -1.

SEE ALSO

sge_conf(5), sge_host_conf(5)

COPYRIGHT

sge_request

NAME

sge_request - Altair Grid Engine default request definition file format

DESCRIPTION

sge_request reflects the format of the files to define default request profiles. If available, default request files are read and processed during job submission before any submit options embedded in the job script and before any options in the qsub(1) or qsh(1) command-line are considered. Thus, the command-line and embedded script options may overwrite the settings in the default request files (see qsub(1) or qsh(1) for details).

There is a cluster global, a user private and a working directory local default request definition file. The working directory local default request file has the highest precedence and is followed by the user private and then the cluster global default request file.

Note, that the -clear option to qsub(1) or qsh(1) can be used to discard any previous settings at any time in a default request file, in the embedded script flags or in a qsub(1) or qsh(1) command-line option.

The format of the default request definition files is:

- The default request files may contain an arbitrary number of lines. Blank lines and lines with a '#' sign in the first column are skipped.
- Each line not to be skipped may contain any qsub(1) option as described in the Altair Grid Engine Reference Manual. More than one option per line is allowed. The batch script file and argument options to the batch script are not considered as qsub(1) options and thus are not allowed in a default request file.

EXAMPLES

The following is a simple example of a default request definition file:

```
=====================================================
# Default Requests File

 # request group to be sun4 and a CPU-time of 5hr
-l arch=sun4,s_cpu=5:0:0

 # don't restart the job in case of system crashes
-r n
=====================================================
```

Having defined a default request definition file like this and submitting a job as follows:

```
qsub test.sh
```

would have precisely the same effect as if the job was submitted with:

```
qsub -l arch=sun4,s_cpu=5:0:0 -r n test.sh
```

FILES

`<sge_root>/<cell>/common/sge_request` - global defaults file
$HOME/.sge_request - user private defaults file
$cwd/.sge_request - cwd directory defaults file

SEE ALSO

sge_intro(1), qsh(1), qsub(1), Altair Grid Engine Installation and Administration Guide

COPYRIGHT

sge_resource_map

NAME

sge_resource_map - Altair Grid Engine resource map configuration file format

DESCRIPTION

A resource map is a special type of complex (see sge_complex(5)) that can be used to manage host resources for which it is required that the job knows which specific resource was assigned (e.g. GPUs). It can only be defined on host level, has complex type **RSMAP** and consists of an integer (amount) and a list of ids. Furthermore a **RSMAP** complex is only allowed to be a consumable **YES**, **JOB** or **HOST** and the relation operator must be "<=". Setting a **default** value is not supported for resource maps.

If the scheduler assigns a specific amount of a RSMAP consumable to a job, it also assigns that amount of ids to the job. The assigned ids can be seen in the qstat -j output and within the environment variables of the job.

FORMAT

Defining RSMAP Complexes

A RSMAP complex is defined in the **complex_values** of a host in the format

```
<name>=<amount>(<id_list>)
```

- `name`: name of the complex
- `amount`: number of available ids
- `id_list`: space separated list of ids. The number of ids must be equal to `amount` and each id has the format

  ```
  <value>:<topology_mask>:<multiplier>[<additional_params>]
  ```

The value of an id is mandatory, can contain any shell compatible characters ([a-z|A-Z|_|0-9]*) and does not need to be unique. The optional `topology_mask` adds core binding features to the id (see description below). `multiplier` is an integer value greater than 0 that shows how often an id is available. A multiplier of 1 does not have any effect on the id and will not be shown in qconf.

`additional_params` is an optional, comma separated list of configuration parameters that are valid for the id they are specified for. Currently the following parameters are supported:

- `amd_id`: See See sge_amd(5)
- `cuda_id`: See sge_nvidia(5)
- `xpu_id`: See See sge_intel(5)
- `device`: Path to one or more devices that are represented by this RSMAP id. Multiple devices can be separated with `|`, e.g. `/dev/nvidia0|/dev/nvidia1`
- `uuid`: See sge_nvidia(5)

`id_list` can also be or contain a numeric range like `1-3` which is equal to following string list: `1 2 3`.

NOTE: If a RSMAP complex is defined without ids it behaves like a **DOUBLE** complex.

NOTE: Altair Grid Engine will always try to compress the list of ids (i.e. reduce the amount of ids by adding numeric ids to ranges and combining identical ids/ranges) to speed up the scheduling and lower the amount of transferred data. The output of qconf -se might not be identical to the ids that were defined with qconf -me/-Me. `<value>::<multiplier>[<additional_params>]` is a valid syntax, but qconf -se will omit the empty topology mask and display only `<value>:<multiplier>[<additional_params>]`.

Requesting RSMAP Complexes

RSMAP complexes can either be requested with only an integer (the scheduler will then assign the next free id(s) to the job) or with an additional expression (see **STRING** in sge_complex(5)). The scheduler will then assign the next free id(s) that match the requested expression. Additional parameters can be added to the request in square brackets. The syntax is:

```
<name>=<amount>[<additional_params>]

or

<name>=<amount>(<expression>)[<additional_params>]
```

Multiple requests for the same complex can be combined with `&`. A job with such a combined request will only be scheduled if all sub-requests can be fulfilled:

```
<name>=<amount>(<expression>)&<amount2>(<expression2>)
```

XOR Operator

RSMAP complexes support the XOR operator (see sge_types(1)). Jobs requesting a RSMAP complex with XOR operator will only get ids assigned that either match the left operand or the right operand, but no combination of them. If no operand is given, the next free id that is available in the requested amount is chosen. The syntax is:

```
<name>=<amount>(^)
```

```
or

<name>=<amount>(<expression>^<expression>)
```

If more than two expressions are used in combination with the XOR operator, all expressions are equally evaluated from left to right. Grouping of XOR expressions within brackets is not supported. There are two different variations of the XOR operator available (for sequential jobs both variations have the same effect):

per Task (`^`): Each task of a job gets only equal ids assigned. Different tasks may get different id(s) assigned.

per Job (`^^`): All tasks of a job get equal id(s) assigned, even if the tasks are running on different hosts.

Examples:

1) Requests two ids of the complex "tape_drive". Either three ids "drive1" or three ids "drive2" need to be available and will be assigned. No combination of different ids is possible.

```
qsub -l tape_drive=3(drive1^drive2) ...
```

2) Requests two ids of the complex "tape_drive" with the **per Job** XOR and wildcards within the expressions. All tasks on all hosts will either get ids matching "drive1*" or id matching "drive2*" assigned (e.g. "drive1 drive1" or "drive1a drive1b" ...).

```
qsub -l tape_drive=2(drive1*^^drive2*) -pe my_pe 2 ...
```

3) Requests two ids of the complex "gpu", both need to have the same value.

```
qsub -l gpu=2(^) ...
```

NOTE: The XOR operator can only be used with RSMAP complexes. When used with other complex types (e.g. **RESTRING**), it will be interpreted as a regular character and has no special effect.

Environment Variables

Jobs with assigned resource maps have set the environment variable `SGE_HGR_<name>=<id_list>`, in order to determine which ids where selected. This is particular useful for devices, where a job needs an access key in order to avoid access collisions by jobs. For PE tasks the environment variable `SGE_HGR_TASK_<name>=<id_list>` is set to determine which ids were selected for the particular task. For a sequential job both environment variables are set. If a resource map is defined on the global host, the corresponding environment variable will be named `SGE_HGR_GLOBAL_<name>`.

Topology Masks

RSMAP ids can be enhanced with topology masks on hosts that support core binding. A topology mask is a string that represents the topology of the current host (see submit(1)) and is appended to each RSMAP id with a colon. If a job gets an id with a topology mask assigned, it will also be bound to the cores selected in the mask (i.e. marked with an uppercase `C`). If not all selected cores are available, a different RSMAP id will be chosen or if no other ids are available, the job will not be scheduled.

Example:

```
complex_values GPU=2(GPU0:SccCC GPU1:SCCcc).
```

When the job requests `-l GPU=1` and the scheduler selects `GPU0`, the job gets bound to the third and fourth core of the host. If `GPU1` is selected, the job is automatically bound to the first and second core of the execution host.

NOTE: In order for this feature to work correctly, the topology masks must have the same amount of sockets and cores as the actual host topology.

Device Mapping

A RSMAP id can be mapped to a device with the optional parameter `device`. If cgroups are enabled and the parameter `devices` is set, Altair Grid Engine will block access to these devices and allow the job only access to the devices that it got assigned via RSMAPs. In addition to the assigned ids, the assigned devices can be seen in qstat -j.

Example:

```
complex_values GPU=2(GPU0[device=/dev/nvidia0] GPU1[device=/dev/nvidia1]).
```

Global Resource Maps

Resource maps can be made globally available by defining them on the global host with `qconf -me global`. Topology masks and other additional parameter (e.g. `device`) can not be configured for global resource maps. Defining resource maps with the same name on the global host and on a local host is currently not supported and the behaviour is undefined.

EXAMPLES

1) Defines a complex “gpu” with two ids “gpu0” and “gpu1” and submits a job that requests one gpu.

```
qconf -me <host>
    ...
    complex_values gpu=2(gpu0 gpu1)


qsub -l gpu=1 ...
```

2) Defines the same complex as in 1), but maps the ids to the NVIDIA GPUs 0 and 1 and their corresponding devices.

```
qconf -me <host>
    ...
    complex_values gpu=2(gpu0[cuda_id=0,device=/dev/nvidia0] \
                         gpu1[cuda_id=1,device=/dev/nvidia1])
```

3) Defines a complex “port” with 65535 ids/ports and submits a job that requests one port, either 443 or 1024.

```
qconf -me <host>
    ...
    complex_values port=65535(1-65535)

qsub -l port=1(443|1024) ...
```

4) Defines a complex “tape_drive” with 2 drives. Each drive and the corresponding id is available twice.

```
qconf -me <host>
    ...
    complex_values tape_drive=4(dev1:2 dev2:2)
```

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), qsub(1), sge_complex(5), sge_host(5), sge_job_class(5).

COPYRIGHT

sge_resource_quota

NAME

sge_resource_quota - Altair Grid Engine resource quota file format

DESCRIPTION

Resource quota sets (RQSs) are a flexible way to set the maximum resource consumption for any job requests. They are used by the scheduler to select the next possible jobs to run. The job request distinction is done by a set of user, project, cluster queue, host, PE and job class filter criteria.

By using resource quota sets, administrators are allowed to define a fine-grained resource quota configuration. This helps restrict some job requests to lesser resource usage, and grants other job requests a higher resource usage.

Note: Jobs requesting an Advance Reservation (AR) are not honored by Resource Quotas, not subject to the resulting limit, and not debited in the usage consumption.

The list of currently configured RQSs can be displayed via the qconf(1) **-srqsl** option. The contents of each listed RQS definition can be shown via the **-srqs** option. The output follows the sge_resource_quota format description. Resource quota sets can be created, modified and deleted via the **-arqs**, **-mrqs** and **-drqs** options to qconf(1).

A resource quota set defines a maximum resource quota for a particular job request. All of the configured rule sets apply all of the time. This means that if multiple resource quota sets are defined, the most restrictive set is used.

Every resource quota set consists of one or more resource quota rules. These rules are evaluated in order, and the first rule that matches a specific request will be used. A resource quota set always results in at most one effective resource quota rule for a specific request.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

A resource quota set definition contains the following parameters:

name

The resource quota set name.

enabled

If set to true the resource quota set is active and will be considered for scheduling decisions. The default value is false.

description

This description field is optional and can be set to an arbitrary string. The default value is NONE.

limit

Every resource quota set needs at least one resource quota rule definition started by the limit field. It's possible to define more resource quota rules divided by a new line. A resource quota rule consists of an optional name, the filters for a specific job request and the resource quota limit.

By default, the expressed limit counts for the entire filter scope. To express a filter-scope-specific limit, it's possible to define an expanded list by setting the list between '{' and '}'. It's only possible to set one complete filter in an expanded list. The tags for expressing a resource quota rule are:

- *name* The name of the rule. The use is optional. The rule name must be unique in one resource quota set.
- *users* Contains a comma-separated list of UNIX users or ACLs (see sge_access_list(5)). This parameter filters for jobs by a user in the list or one of the ACLs in the list. Any user not in the list will not be considered for the resource quota rule. The default value is '*', which means any user. An ACL is differentiated from a UNIX user name by prefixing the ACL name with an'@' sign. To exclude a user or ACL from the rule, the name can be prefixed with the '!' sign. Defined UNIX user or ACL names need not be known in the Altair Grid Engine configuration.
- *projects* Contains a comma-separated list of projects (see sge_project(5)). This parameter filters for jobs requesting a project in the list. Any project not in the list will not be considered for the resource quota rule. If no "projects" filter is specified, all projects and jobs with no requested project match the rule. The value '' *means all jobs with requested projects. To exclude a project from the rule, the name can be prefixed with the '!' sign. The value '!'* means only jobs with no project requested.
- *pes* Contains a comma-separated list of PEs (see sge_pe(5)). This parameter filters for jobs requesting a PE in the list. Any PE not in the list will not be considered for the resource quota rule. If no "pes" filter is specified, all PEs and jobs with no requested PE match the rule. The value '' *means all jobs with requested PEs. To exclude a PE from the rule, the name can be prefixed with the '!' sign. The value '!'* means only jobs with no PE requested.
- *jcs* Contains a comma-separated list of job class or job class variant names (see sge_job_class(5) and **jc_list** in sge_types(1)). This parameter filters for jobs requesting

a job class variant in the list. If the variant part of a name is omitted then the **default** variant is used as the filter. Any job class not in the list will not be considered for the resource quota rule. If no "jcs" filter is specified, all variants of all job classes and jobs with no job class specification match the rule. To exclude a job class variant from the rule, the name can be prefixed with the exclamation mark (!). '!*' means only jobs that were not derived from a job class.

- *queues* Contains a comma-separated list of cluster queues (see sge_queue_conf(5)). This parameter filters for jobs that may be scheduled in a queue in the list. Any queue not in the list will not be considered for the resource quota rule. The default value is '*', which means any queue. To exclude a queue from the rule, the name can be prefixed with the'!' sign.
- *hosts* Contains a comma-separated list of host or hostgroups (see host(1) and sge_hostgroup(5)). This parameter filters for jobs that may be scheduled in a host in the list or a host contained in a hostgroup in the list. Any host not in the list will not be considered for the resource quota rule. The default value is '*', which means any hosts. To exclude a host or hostgroup from the rule, the name can be prefixed with the'!' sign.
- *to* This mandatory field defines the quota for resource attributes for this rule. The quota is expressed by one or more limit definitions, separated by commas. There are two kinds of limit definitions:
- *static limits* Static limits set static values as quotas. Each limit consists of a complex attribute followed by an "=" sign and the value specification compliant with the complex attribute type (see sge_complex(5)).
- *dynamic limits* A dynamic limit is a simple algebraic expression used to derive the limit value. To be dynamic, the formula can reference a complex attribute whose value is used for the calculation of the resulting limit. The formula expression syntax is that of a sum of weighted complex values, that is:

  {w1|$complex1[*w1]}[{+|-}{w2|$complex2[*w2]}[{+|-}...]]

The weighting factors (w1, ...) are positive integers or floating point numbers in double precision. The complex values (complex1, ...) are specified by the name defined as type INT or DOUBLE in the complex list (see sge_complex(5)).
Note: Dynamic limits can only configured for a host-specific rule.

Please note that resource quotas are not enforced as job resource limits. Limiting for example h_vmem in a resource quota set does not result in a memory limit being set for job execution.

EXAMPLES

The following is the simplest form of a resource quota set. It restricts all users together to the maximal use of 100 slots in the whole cluster.

```
==============================================================================
{
name max_u_slots
description "All users max use of 100 slots"
enabled true
limit to slots=100
}
==============================================================================
```

The next example restricts user1 and user2 to 6g virtual_free and all other users to the maximal use of 4g virtual_free on every host in hostgroup lx_hosts.

```
==============================================================================
{
name max_virtual_free_on_lx_hosts
description "resource quota for virtual_free restriction"
enabled true
limit users {user1,user2} hosts {@lx_host} to virtual_free=6g
limit users {*} hosts {@lx_host} to virtual_free=4g
}
==============================================================================
```

The next example shows the use of a dynamic limit. It restricts all users together to a maximum use of double the size of num_proc.

```
==============================================================================
{
name max_slots_on_every_host
enabled true
limit hosts {*} to slots=$num_proc*2
}
==============================================================================
```

SEE ALSO

sge_intro(1), sge_access_list(5), sge_complex(5), sge_host_conf(5), sge_hostgroup(5), qconf(1), qquota(1), sge_project(5), sge_job_class(5).

COPYRIGHT

sge_sched_conf

NAME

sched_conf - Altair Grid Engine default scheduler configuration file

DESCRIPTION

sched_conf defines the configuration file format for Altair Grid Engine's scheduler. In order to modify the configuration, use the graphical user's interface qmon(1) or the -msconf option of the qconf(1) command. A default configuration is provided together with the Altair Grid Engine distribution package.

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

The following parameters are recognized by the Altair Grid Engine scheduler if present in sched_conf:

algorithm

Note: Deprecated, may be removed in future release.
Allows for the selection of alternative scheduling algorithms.

Currently **default** is the only allowed setting.

host_sort_formula

A algebraic expression used to derive a single weighted load value from all or part of the load parameters reported by sge_execd(8) for each host and from all or part of the consumable resources (see sge_complex(5)) being maintained for each host. The load formula expression syntax is that of a summation weighted load values, that is:

```
host_sort_formula := weighted_value [ operator weighted_value ] .
operator := â+â | â-â .
weighted_value := weight | <load_value> [ â*â weight ] .
weight := <positive_integer> .
```

Note, no blanks are allowed in the load formula.

<load_value> represents a load value (see sge_execd(8)) or consumable resource being maintained for each host (see sge_complex(5)) or an administrator defined load value (see the load_sensor parameter in sge_conf(5)).

<positive_integer> represents a positive integer value.

The default load formula is "np_load_avg".

job_load_adjustments

The load, which is imposed by the Altair Grid Engine jobs running on a system varies in time, and often, e.g. for the CPU load, requires some amount of time to be reported in the appropriate quantity by the operating system. Consequently, if a job was started very recently, the reported load may not provide a sufficient representation of the load which is already imposed on that host by the job. The reported load will adapt to the real load over time, but the period of time, in which the reported load is too low, may already lead to an oversubscription of that host. Altair Grid Engine allows the administrator to specify **job_load_adjustments** which are used in the Altair Grid Engine scheduler to compensate for this problem.
The **job_load_adjustments** are specified as a comma separated list of arbitrary load parameters or consumable resources and (separated by an equal sign) an associated load correction value. Whenever a job is dispatched to a host by the scheduler, the load parameter and consumable value set of that host is increased by the values provided in the **job_load_adjustments** list. These correction values are decayed linearly over time until after **load_adjustment_decay_time** from the start the corrections reach the value 0. If the **job_load_adjustments** list is assigned the special denominator NONE, no load corrections are performed.
The adjusted load and consumable values are used to compute the combined and weighted load of the hosts with the **host_sort_formula** (see above) and to compare the load and consumable values against the load threshold lists defined in the queue configurations (see sge_queue_conf(5)). If the **host_sort_formula** consists simply of the default CPU load average parameter np_load_avg, and if the jobs are very compute intensive, one might want to set the **job_load_adjustments** list to np_load_avg=1.00, which means that every new job dispatched to a host will require 100 % CPU time, and thus the machine's load is instantly increased by 1.00.

load_adjustment_decay_time

The load corrections in the "**job_load_adjustments**" list above are decayed linearly over time from the point of the job start, where the corresponding load or consumable parameter is raised by the full correction value, until after a time period of "**load_adjustment_decay_time**", where the correction becomes 0. Proper values for "**load_adjustment_decay_time**" greatly depend upon the load or consumable parameters used and the specific operating system(s). Therefore, they can only be determined on-site and experimentally. For the default np_load_avg load parameter a "**load_adjustment_decay_time**" of 7 minutes has proven to yield reasonable results.

maxujobs

The maximum number of jobs any user may have running in a Altair Grid Engine cluster at the same time. If set to 0 (default) the users may run an arbitrary number of jobs.

schedule_interval

At the time the scheduler thread initially registers at the event master thread in sge_qmaster(8)process **schedule_interval** is used to set the time interval in which the event master thread sends scheduling event updates to the scheduler thread. A scheduling event is a status change that has occurred within sge_qmaster(8) which may trigger or affect scheduler decisions (e.g. a job has finished and thus the allocated resources are available again).
In the Altair Grid Engine default scheduler the arrival of a scheduling event report triggers a scheduler run. The scheduler waits for event reports otherwise.**
Schedule_interval** is a time value (see sge_queue_conf(5) for a definition of the syntax of time values).

halftime

When executing under a share based policy, the scheduler "ages" (i.e. decreases) usage to implement a sliding window for achieving the share entitlements as defined by the share tree. The **halftime** defines the time interval in which accumulated usage will have been decayed to half its original value. Valid values are specified in hours or according to the time format as specified in sge_queue_conf(5).
If the value is set to 0, the usage is not decayed. -1 results in immediate decay.

usage_weight_list

Altair Grid Engine accounts for the consumption of resources to determine the usage which is imposed on the system by a job. A single usage value is computed from built-in usage values and consumable complex values by multiplying the individual values by weights and adding them up. The weights are defined in the **usage_weight_list**. The format of the list is

```
<name>=<weight>,<name2>=<weight2>,...
```

where **<name>** is either the name of a built-in usage value (**wallclock**, **cpu**, **mem**, **io**) or of any consumable complex and **<weight>** is the corresponding weight. The weights are real numbers and the sum of all tree weights should be 1.

If a consumable has the same name as one of the built-in usage values, Altair Grid Engine will always use the built-in usage value and omit the value of the consumable.

compensation_factor

Determines how fast Altair Grid Engine should compensate for past usage below of above the share entitlement defined in the share tree. Recommended values are between 2 and 10, where lower values mean faster compensation. This means that higher values limit the compensation for share tree nodes which are below their target. It will take longer time until these nodes will reach their targeted shares.

weight_user

The relative importance of the user shares in the functional policy. Values are of type real.

weight_project

The relative importance of the project shares in the functional policy. Values are of type real.

weight_department

The relative importance of the department shares in the functional policy. Values are of type real.

weight_job

The relative importance of the job shares in the functional policy. Values are of type real.

weight_tickets_functional

The maximum number of functional tickets available for distribution by Altair Grid Engine. Determines the relative importance of the functional policy. See under sge_priority(5) for an overview on job priorities.

weight_tickets_share

The maximum number of share based tickets available for distribution by Altair Grid Engine. Determines the relative importance of the share tree policy. See under sge_priority(5) for an overview on job priorities.

weight_deadline

The weight applied on the remaining time until a jobs latest start time. Determines the relative importance of the deadline. See under sge_priority(5) for an overview on job priorities.

weight_waiting_time

The weight applied on the jobs waiting time since submission. Determines the relative importance of the waiting time. See under sge_priority(5) for an overview on job priorities.

weight_urgency

The weight applied on jobs normalized urgency when determining priority finally used. Determines the relative importance of urgency. See under sge_priority(5) for an overview on job priorities.

weight_priority

The weight applied on jobs normalized POSIX priority when determining priority finally used. Determines the relative importance of POSIX priority. See under sge_priority(5) for an overview on job priorities.

weight_ticket

The weight applied on normalized ticket amount when determining priority finally used. Determines the relative importance of the ticket policies. See under sge_priority(5) for an overview on job priorities.

weight_queue_host_sort

Find more details in the section for parameter weight_host_affinity below.

weight_queue_affinity

Find more details in the section for parameter weight_host_affinity below.

weight_queue_seqno

Find more details in the section for parameter weight_host_affinity below.

weight_host_sort

Find more details in the section for parameter weight_host_affinity below.

weight_host_affinity

Dispatching of jobs works on a sorted queue list. Queue sorting is based on a host sort value (influenced by host load and host affinity), queue sequence number and queue affinity.

Weighting factors for each of the three sources (**weight_queue_host_sort**, **weight_queue_affinity** and **weight_queue_seqno**) allow to define the influence on the final sort value for queue instances and therefore allow to influence which queues are used first if multiple queues would provide enough resources to dispatch a job.

The host sort value is calculated from the normalized host load according to the **host_sort_formula** and the host affinity.

The influence between the two sources on the host sort can be weighted with **weight_host_sort** and **weight_host_affinity**.

flush_finish_sec

The parameters are provided for tuning the system's scheduling behavior. By default, a scheduler run is triggered in the scheduler interval. When this parameter is set to 1 or larger, the scheduler will be triggered x seconds after a job has finished. Setting this parameter to 0 disables the flush after a job has finished.

flush_submit_sec

The parameters are provided for tuning the system's scheduling behavior. By default, a scheduler run is triggered in the scheduler interval. When this parameter is set to 1 or larger, the scheduler will be triggered x seconds after a job was submitted to the system. Setting this parameter to 0 disables the flush after a job was submitted.

schedd_job_info

The default scheduler can keep track why jobs could not be scheduled during the last scheduler run. This parameter enables or disables the observation. The value **true** enables the monitoring **false** turns it off.

It is also possible to activate the observation only for certain jobs by setting this parameter to **if_requested** and by using the **-rdi** submit switch to enable collection for a specific job and all other jobs that belong to the same job category.

The user can obtain the collected information with the command qstat -j.

params

This is foreseen for passing additional parameters to the Altair Grid Engine scheduler. The following values are recognized:

- *DURATION_OFFSET* If set, overrides the default of value 60 seconds. This parameter is used by the Altair Grid Engine scheduler when planning resource utilization as the delta between net job runtimes and total time until resources become available again. Net job runtime as specified with -l h_rt=… or -l s_rt=… or -l d_rt=… or **default_duration** always differs from total job runtime due to delays before and after actual job start and finish. Among the delays before job start is the time until the end of a **schedule_interval**, the time it takes to deliver a job to sge_execd(8) and the delays caused by **prolog** in sge_queue_conf(5) , **start_proc_args** in sge_pe(5) and **starter_method** in sge_queue_conf(5) (**notify**, **terminate_method** or **checkpointing**), procedures run after actual job finish, such as **stop_proc_args** in sge_pe(5) or **epilog** in sge_queue_conf(5) , and the delay until a new **schedule_interval**.
  If the offset is too low, resource reservations (see **max_reservation**) can be delayed repeatedly due to an overly optimistic job circulation time.

- *PROFILE* If set equal to 1, the scheduler logs profiling information summarizing each scheduling run. In combination with WARN_DISPATCHING_TIME it is possible to get profiling data for the longest and shortest job scheduling. The man page sge_diagnostics(5) shows detailed information about scheduler thread profiling.

- *ENABLE_PRIORITY_ORDER* Do not use in productive environments. If set equal to 1, the scheduler will send internal GDI order requests as priority requests.

- *MONITOR* If set equal to 1, the scheduler records information for each scheduling run allowing to reproduce job resources utilization in the file `<sge_root>/<cell>/common/schedule`.
  In order to see entries in the schedule file resource reservation must be turned on (**max_reservation** must be greater than 0) and jobs need a run-time (using h_rt, s_rt, d_rt or setting a **default_duration**).
  The format of the schedule file is:

  : The jobs id.
  : The array task id or 1 in case of non-array jobs.
  : One of RUNNING, SUSPENDED, MIGRATING, STARTING, RESERVING.
  : Start time in seconds after 1.1.1970.
  : Assumed job duration in seconds.
  : One of {P, G, H, Q} standing for {PE, Global, Host, Queue}.
  : The name of the PE, global, host, queue.
  : The name of the consumable resource.
  The resource utilization debited for the job.
  A line ":::::::::" marks the begin of a new schedule interval.

Please note this file is not truncated. Make sure the monitoring is switched off in case you have no automated procedure setup that truncates the schedule file.

- *MONITOR_CALENDAR* If set equal to 1, the information being logged via the MONITOR switch is extended by calendar events. For the two next calendar periods of a queue instance a record is written to the schedule file.

- *PE_RANGE_ALG* This parameter sets the algorithm for the pe range computation. The default is "bin", which means that the scheduler will use a binary search to select the best one. It should not be necessary to change it to a different setting in normal operation. If a custom setting is needed, the following values are available:

- auto : the scheduler selects the best algorithm
- least : starts the resource matching with the lowest slot amount first
- bin : starts the resource matching in the middle of the pe slot range
- highest : starts the resource matching with the highest slot amount first
- *PREFER_SOFT_REQUESTS* If this parameter is set scheduler will try to find an assignment or a resource reservation which matches as many soft requests as possible. "PREFER_SOFT_REQUESTS" only has impact on parallel jobs. In case of the dispatching of jobs (no reservation) by default (PREFER_SOFT_REQUESTS not set) resources will be preferred which provide more slots (in case of pe ranges), with the parameter set resources will be preferred which have less infringements for soft requests. In case of resource reservation without the parameter set the scheduler reserves the earliest available resources in time even when soft requests for the job can not be fulfilled. When the parameter is set then resources are preferred which have less infringements for soft requests.
- *PE_SORT_ORDER* When using wildcard parallel environment selection during submission time, the parallel environment the scheduler chooses is arbitrary. In order to fill up the parallel environments in a specific order this parameter allows to change the sorting of matching parallel environments either to an ascending or descending order. When PE_SORT_ORDER is set to ASCENDING (or 1) the first PE which is tested for job selection is the one which is in alpha-numerical order the first one (test1pe before test2pe and test10pe before test2pe, when submitting with -pe test*). When it is set to DESCENDING (or 2) the PE which is tested is in alpha-numerical order the last one (testpe2 in the previous example). When PE_SORT_ORDER is set to CAPACITY (or 3) the list of parallel environments is sorted by free slots descending for job dispatching and by earliest time without an existing reservation for resource reservation. PE_SORT_ORDER=CAPACITY makes most sense when combined with PE_ACCEPT_FIRST_ASSIGNMENT. When it is set to 0 or NONE then the first matching PE is arbitrary (default), which is a good choice for balancing PEs and the same than with absence of the parameter.
- *PE_ACCEPT_FIRST_ASSIGNMENT* When a job is submitted with wildcard pe request the scheduler will try to find the best matching pe. When PE_ACCEPT_FIRST_ASSIGNMENT is set to 1 or TRUE then the first matching assignment will be selected. This can significantly improve scheduling times when wildcard pe requests are used but can lead to less optimal assignments, e.g. the chosen assignment may provide a lower amount of slots from pe slot ranges or might match less soft requests. Default value for PE_ACCEPT_FIRST_ASSIGNMENT is 0 / FALSE. For resource reservation scheduling PE_ACCEPT_FIRST_ASSIGNMENT only is used when combined with PE_SORT_ORDER=CAPACITY.
- *PE_ACCEPT_ASSIGNMENT_TIME* When a job is submitted with wildcard pe request the scheduler will try to find the best matching pe. When PE_ACCEPT_ASSIGNMENT_TIME is set to a value > 0 (seconds with decimal places, e.g. 0.01) then scheduler will stop searching for better assignments once an assignment has been found and

the time given by PE_ACCEPT_ASSIGNMENT_TIME is exceeded. This can significantly improve scheduling times when wildcard pe requests are used but can lead to less optimal assignments, e.g. the chosen assignment may provide a lower amount of slots from pe slot ranges or might match less soft requests. Default value for PE_ACCEPT_ASSIGNMENT_TIME is 0 (feature disabled). PE_ACCEPT_ASSIGNMENT_TIME has no effect on resource reservation scheduling.

- *PE_ACCEPT_SOFT_VIOLATIONS* When a job is submitted with wildcard pe request the scheduler will try to find the best matching pe, which means the pe with the maximum possible number of slots and the least number of soft violations (soft requests which cannot be matched). As long as the job assignments found so far still have soft violations scheduler will continue looking for better assignments until it either found an assignment with 0 soft violations or it reached the end of the list of matching pes. When PE_ACCEPT_SOFT_VIOLATIONS is set to a number > 0 scheduler will accept assignments having up to the given amount of soft violations (soft requests not being matched). Default value for PE_ACCEPT_SOFT_VIOLATIONS is 0 (feature disabled). PE_ACCEPT_SOFT_VIOLATIONS has no effect on resource reservation scheduling.

- *COUNT_CORES_AS_THREADS* If set to 1 or TRUE the scheduler treats the requested number of cores of a job (with -binding parameter) as request for hardware supported threads. On hosts with SMT (topology string with threads, like SCTTCTT) the number of requested cores is divided by the number of threads per core. In case a core is filled only partially the complete core is requested by the job. Example: When a job requests 3 cores, on a host with hyper-threading (2 hardware threads per core) the request is transformed to 2 cores (because 3 threads are needed). On a host without hyper-threading the job requests 3 cores, and on a host with 4 hardware-threads supported per core the job requests 1 core.

- *SKIP_HOSTS_WITH_NO_FREE_CORES* If set to 1 or TRUE, the scheduler will skip hosts that have no free cores (i.e. all cores bound by jobs using -binding), for all jobs, even for those that do not request a core-binding themselves.

- *WRITE_SCHEDD_RUNLOG* If set equal to 1, scheduler will write trace messages of the next scheduling run to the file `<sge_root>/<cell>/common/schedd_runlog` when triggered by qconf -tsm. Writing the schedd_runlog file can have significant impact on scheduler performance. This feature should only be enabled when the debugging information contained in the file is actually needed. Default setting is disabled.

- *MAX_SCHEDULING_TIME* This parameter can be used to specify a maximum time interval (time_specifier, see sge_types(1)) for one scheduling run. If the scheduler has not finished a dispatching run within this time interval job dispatching is stopped for this one scheduling run. In the next scheduling run job dispatching again starts with the highest priority job. Default for this parameter is 0 (do full dispatching from the highest priority job down to the lowest priority job). In huge clusters with a high number of pending jobs setting this parameter to reasonable values (e.g. one minute) can improve cluster utilization and responsiveness of sge_qmaster.

- *MAX_DISPATCHED_JOBS* This parameter can be used to limit the number of jobs which get scheduled in one scheduling interval. Can be set to any positive number or 0 (do not limit the number of scheduled jobs). Default is 0. Limiting the number of jobs getting scheduled in a single scheduling interval can be useful to avoid overload on

the cluster, especially on file servers due to many jobs starting up at the same time. But use this option with care: Setting it to a too low value can lead to bad utilization of the cluster.

- *HIGH_PRIO_DRAINS_CLUSTER* When this parameter is set to 1 or TRUE the cluster will be drained until the highest priority job could be scheduled. This can be used as a workaround to avoid starvation of parallel jobs when resource reservation cannot be applied, e.g. as job runtimes are unknown. Use this parameter with care and only temporarily: It can lead to very bad utilization of the cluster.
- *WARN_DISPATCHING_TIME* When this parameter is set to a threshold in milliseconds the Altair Grid Engine scheduler will print a warning to the sge_qmaster(8) messages file when dispatching a job takes longer than the given threshold. If this parameter is enabled and PROFILE is turned on the profiling output will contain additional information about the longest and shortest job scheduling time. The default for "*WARN_DISPATCHING_TIME*" is 0 (switched off).
- *SHARE_BASED_ON_SLOTS* When this parameter is set to 1 or TRUE, the scheduler will consider the number of slots being used by running jobs and by pending jobs when pushing users and projects toward their sharing targets as defined by the share tree. That is, a parallel job using 4 slots will be considered to be equal to 4 serial jobs. When the parameter is set to FALSE (default), every job is considered equal. The **urgency_slots** PE attribute in sge_pe(5) will be used to determine the number of slots when a job is submitted with a PE range.
- *MAX_PENDING_REASONS* This parameter can be used to limit the total number of messages of why jobs could not be scheduled during a scheduling interval when schedd_job_info is set to something other than false. This limit exists to prevent the *sge_qmaster(8)* from consuming too much memory. The default value of 100,000 should be adequate for most systems. Once the limit is reached, a message indicating that the maximum limit has been reached is added and no additional messages will be collected for that scheduling interval. For this reason, it is recommended to adjust **MAX_SIMILAR_PENDING_REASONS** and **MAX_SIMILAR_GLOBAL_PENDING_REASONS** to reasonable values rather than using **MAX_PENDING_REASONS**.
- *MAX_SIMILAR_PENDING_REASONS* This parameter can be used to limit the total number of similar messages of why each job could not be scheduled during a scheduling interval when schedd_job_info is set to something other than false. Complex scheduler configurations (such as many execution hosts, queues, and parallel environments) can result in a huge number of specific but similar reasons why each job could not be scheduled. This limit exists to prevent the sge_qmaster from consuming too much memory and to reduce the number of similar messages to a reasonable amount which can effectively represent why the jobs cannot be scheduled. The default value of 1,000 should be adequate for most systems. However, this number can be reduced or increased to show fewer or more of the similar messages. It is recommended to adjust this value rather than **MAX_PENDING_REASONS** since it will better represent the reasons why each job cannot be scheduled.
- *MAX_SIMILAR_GLOBAL_PENDING_REASONS* This parameter can be used to limit the total number of similar global (non-job-specific) messages of why jobs could not be scheduled during a scheduling interval when schedd_job_info is set to something other than

false. Complex scheduler configurations (such as many execution hosts, queues, and parallel environments) can result in a huge number of similar common reasons why jobs could not be scheduled. This limit exists to prevent the sge_qmaster from consuming too much memory and to reduce the number of similar messages to a reasonable amount which can effectively represent why jobs cannot be scheduled. The default value of 1,000 should be adequate for most systems. However, this number can be reduced or increased to show fewer or more of the similar messages. It is recommended to adjust this value rather than **MAX_PENDING_REASONS** since it will better represent the reasons why jobs cannot be scheduled.

- *FIND_EARLIEST_RESERVATION_PER_PE* This parameter affects the reservation of resources. When the scheduler does reservation scheduling it tries to find a reservation time before the end of running jobs and other reservations. Once it finds a possible reservation time it will select this time. There might be earlier possible times, though, e.g. in gaps between a running job and a calendar closing the queue. When the scheduler param *FIND_EARLIEST_RESERVATION_PER_PE* is set to "TRUE" or "1" then scheduler will try to find such gaps and do the reservation at the earliest time possible. Default for FIND_EARLIEST_RESERVATION_PER_PE is "FALSE". Be aware that setting the parameter can have a performance impact. It might increase scheduling times, especially in scenarios with many reservations enabled and with queue calendars configured.

Changing **params** will take immediate effect. The default for **params** is none.

reprioritize_interval

Interval (HH:MM:SS) to reprioritize jobs on the execution hosts based on the current ticket amount for the running jobs. If the interval is set to 00:00:00 the reprioritization is turned off. The default value is 00:00:00. The reprioritization tickets are calculated by the scheduler and update events for running jobs are only sent after the scheduler calculated new values. How often the schedule should calculate the tickets is defined by the reprioritize_interval. Because the scheduler is only triggered in a specific interval (scheduler_interval) this means the reprioritize_interval has only a meaning if set greater than the scheduler_interval. For example, if the scheduler_interval is 2 minutes and reprioritize_interval is set to 10 seconds, this means the jobs get re-prioritized every 2 minutes.

report_pjob_tickets

This parameter allows to tune the system's scheduling run time. It is used to enable / disable the reporting of pending job tickets to the qmaster. It does not influence the tickets calculation. The sort order of jobs in qstat and qmon is only based on the submit time, when the reporting is turned off.
The reporting should be turned off in a system with a very large amount of jobs by setting this parameter to "false".

report_rjob_tickets

This parameter allows to tune the system's scheduling run time. It is used to enable / disable the reporting of running job tickets to the qmaster. It does not influence the tickets calculation. The sort order of jobs in qstat and qmon is only based on the submit time, when the reporting is turned off.
The reporting should be turned off in a system with a very large amount of jobs by setting this parameter to "false".

halflife_decay_list

The halflife_decay_list allows to configure different decay rates for the finished_jobs usage types, which is used in the pending job ticket calculation to account for jobs which have just ended. This allows the user the pending jobs algorithm to count finished jobs against a user or project for a configurable decayed time period. This feature is turned off by default, and the halftime is used instead.
The halflife_decay_list also allows one to configure different decay rates for each usage type being tracked (cpu, io, and mem). The list is specified in the following format:

```
<USAGE_TYPE>=<TIME>[:<USAGE_TYPE>=<TIME>[:<USAGE_TYPE>=<TIME>]]
```

Usage_TYPE> can be one of the following: cpu, io, or mem.
<TIME> can be -1, 0 or a timespan specified in minutes. If <TIME> is -1, only the usage of currently running jobs is used. 0 means that the usage is not decayed.

policy_hierarchy

This parameter sets up a dependency chain of ticket based policies. Each ticket based policy in the dependency chain is influenced by the previous policies and influences the following policies. A typical scenario is to assign precedence for the override policy over the share-based policy. The override policy determines in such a case how share-based tickets are assigned among jobs of the same user or project. Note that all policies contribute to the ticket amount assigned to a particular job regardless of the policy hierarchy definition. Yet the tickets calculated in each of the policies can be different depending on "POLICY_HIERARCHY".

The "POLICY_HIERARCHY" parameter can be a up to 3 letter combination of the first letters of the 3 ticket based policies S(hare-based), F(unctional) and O(verride). So a value "OFS" means that the override policy takes precedence over the functional policy, which finally influences the share-based policy. Less than 3 letters mean that some of the policies do not influence other policies and also are not influenced by other policies. So a value of "FS" means that the functional policy influences the share-based policy and that there is no interference with the other policies.

The special value "NONE" switches off policy hierarchies.

share_override_tickets

If set to “true” or “1”, override tickets of any override object instance are shared equally among all running jobs associated with the object. The pending jobs will get as many override tickets, as they would have, when they were running. If set to “false” or “0”, each job gets the full value of the override tickets associated with the object. The default value is “true”.

share_functional_shares

If set to “true” or “1”, functional shares of any functional object instance are shared among all the jobs associated with the object. If set to “false” or “0”, each job associated with a functional object, gets the full functional shares of that object. The default value is “true”.

max_functional_jobs_to_schedule

The maximum number of pending jobs to schedule in the functional policy. The default value is 200.

max_pending_tasks_per_job

The maximum number of subtasks per pending array job to schedule. This parameter exists in order to reduce scheduling overhead. The default value is 50.

fair_urgency_list

A list of complex attributes for which fair urgency shall be applied.

Without fair urgency every job requesting a resource having urgency gets the full urgency assigned. With fair urgency the first job requesting a resource gets the full urgency, the second job gets half of the urgency, the third job a third of the urgency ...

This influences the sorting of the pending job list and can be used to get an even distribution of jobs across multiple resources.

max_reservation

The maximum number of reservations scheduled within a schedule interval. When a runnable job can not be started due to a shortage of resources a reservation can be scheduled instead. A reservation can cover consumable resources with the global host, any execution host and any queue. For parallel jobs reservations are done also for slots resource as specified in sge_pe(5). As job runtime the maximum of the time specified with -l h_rt=... or -l s_rt=... or -l d_rt=... is assumed. For jobs that have neither of them the default_duration is assumed. Reservations prevent jobs of lower priority as specified in *sge_priority*(5) from utilizing the reserved resource quota during the time of reservation.

Jobs of lower priority are allowed to utilize those reserved resources only if their prospective job end is before the start of the reservation (backfilling). Reservation is done only for non-immediate jobs (-now no) that request reservation (-R y). If max_reservation is set to "0" no job reservation is done.

Note, that reservation scheduling can be performance consuming and hence reservation scheduling is switched off by default. Since reservation scheduling performance consumption is known to grow with the number of pending jobs, the use of -R y option is recommended only for those jobs actually queuing for bottleneck resources. Together with the max_reservation parameter this technique can be used to narrow down performance impacts.

default_duration

When job reservation is enabled through max_reservation *sge_sched_conf*(5) parameter the default duration is assumed as runtime for jobs that have neither -l h_rt=... nor -l s_rt=... nor -l d_rt=... specified. In contrast to a h_rt/s_rt time limit the d_rt and the default_duration are not enforced.

backfilling

When job reservation is enabled through the max_reservation *sge_sched_conf*(5) parameter jobs fitting before resource reservations can be backfilled. Backfilling requires a job runtime specification. If a job does not request a runtime via the h_rt, s_rt or d_rt attribute the default duration is assumed as runtime. Using default duration or a badly estimated d_rt runtime can lead to false backfilling decisions, therefore the **backfilling** parameter allows switching off or limiting the scope of backfilling. It can be set to one the following values:

Value	Description
OFF	Scheduler will never do backfilling.
H_RT	Only jobs requesting a runtime via the h_rt limit can be backfilled.
ON	Backfilling is enabled for all jobs types (default).

prioritize_preemptees

When preemptive scheduling is enabled and when this parameter is set to **TRUE** then the scheduler will create a reservation for preemptees before the regular scheduling run is done. This ensures that preemptees get started again at least when the preemptor finishes, unless resources required by the preemptee are still held by jobs which got backfilled. **prioritize_preemptees** in combination with disabling of backfilling (by setting **backfilling** to **OFF**) provides a guarantee that preemptees get restarted at least when the preemptor finishes, at the expense of lower cluster utilization. Default for this parameter is **FALSE**.

preemptees_keep_resources

When this parameter is set to **TRUE** then jobs that get preempted will only be enforced to free those resources that will be consumed by the job (preemptor) that causes the preemption. This prevents resources of a preemptee from getting consumed by other jobs. **preemptees_keep_resources** and **prioritize_preemptees** in combination provides a guarantee that preemptees get restarted at latest when the preemptor finishes, at the expense of a waste of resources and a bad cluster utilization. One exception from this are software licenses managed through Altair License Orchestrator. Those resources cannot be held by a preemptee because the preemptee process will be suspended and the underlying license manager might assume the license to be free anyways. Default for this parameter is **FALSE**.

max_preemptees

Defines the maximum number of preemptees in the cluster. As preempted jobs might hold some resources (memory) and through the **preemptees_keep_resources** parameter might even hold most of their resources a high number of preemptees can significantly impact cluster operation. Limiting the number of preemptees will limit the amount of held but unused resources. Default for this parameter is 0.

preemption_distance

A preemption will only be triggered if the resource reservation that could be done for a job is farther in the future than the given time interval (hh:mm:ss). Reservation can be disabled by setting the value to **00:00:00**. Reservation will also be omitted if preemption of jobs is forced manually using 'qmod -f -p ... S|N|P'. Default for this parameter is **00:15:00**.

preemption_priority_adjustments

This parameter allows to automatically adjust the POSIX priority of running jobs depending on their type or state. This will influence the normalized and weighted priority (prio as show by qstat) before running jobs are considered as preemption candidates. As default this parameter is set to **NONE** but it is allowed to set it to a list of name/value-pairs. The name of such an entry defines a possible type, state or other characteristic of a running job and the value defines the new POSIX priority or a relative POSIX priority adjustment.

Name/value-pairs have to be separated by comma (','). Delimiting character for name and value is the equal-character ('='). Priority values that would leave the allowed POSIX priority range will be automatically set to the smallest or biggest priority value depending on the limit that is exceeded.

Adjustment value might be in range from -1023 to 1024. Relative values start with the letter 'd' (for delta) and have to be in range -2047 to 2047 (e.g 'd-100' to decrease the POSIX priority by 100).

Please note that currently the list may only contain one name/value pair but this may change with each patch release of Altair Grid Engine. If the list contains multiple entries then all of them are considered from left to right (first to last entry).

ALREADY_PREEMPTED Adjusts the priority of jobs that have been restarted after preemption. Prevent that jobs that have been restarted after preemption get immediately preempted again by a higher priority job.

FILES

`<sge_root>/<cell>/common/sched_configuration` - scheduler thread configuration

SEE ALSO

sge_intro(1), qalter(1), qconf(1), qstat(1), qsub(1), sge_complex(5), sge_queue_conf(5), sge_execd(8), sge_qmaster(8), Altair Grid Engine Installation and Administration Guide

COPYRIGHT

sge_session_conf

NAME

Sessions - Altair Grid Engine session configuration

DESCRIPTION

When Altair Grid Engine client commands interact with Altair Grid Engine server components (see sge_qmaster(8)) then this is done by using an interface named GDI (Grid Engine Data Interface). This interface is used to send client requests to the Altair Grid Engine system that are then handled within the server component and answered by a response message that contains the result for the client request.

This GDI interface is also used for internal Altair Grid Engine communication between components running on execution hosts (see sge_execd(8)) as well as for internal communication between components within the sge_qmaster(8) component itself.

GDI requests can be divided into two categories: Requests that will change the configuration/state of the Altair Grid Engine system (read-write-requests) and requests that will gather information to display the configuration/state of the Altair Grid Engine system (read-only-requests).

Altair Grid Engine 8.2 has been redesigned so that read-write-requests and read-only-requests can be executed completely independently from each other. Furthermore up to 64 read-only requests can work in parallel which is not possible in Sun Grid Engine, Oracle Grid Engine and other open source versions of Grid Engine. This ensures faster response times for all requests and has a huge positive impact on the cluster throughput.

The drawback of this approach is that GDI-read-only requests might not see the outcome of recently executed read-write requests in certain situations. E.g. it might happen that a user submits a job (read-write-request) and immediately does a qstat -j <jid> (read-only-request) which responds with an error which says that the previously created job does not exist.

In some cases such behavior may cause problems and it is desired that requests should be executed in sequence and for this reason GDI sessions have been introduced that guarantee a consistent view onto the Altair Grid Engine system. Internally read-only requests that are executed within the control of a session are delayed until they can see all changes that have happened previously within the same session. The maximum delay depends on the Altair Grid Engine system load and the number of threads that are active. This value also can be influenced by the *max_reader_delay* parameter which can be defined as *qmaster_param* in the Altair Grid Engine global configuration (see sge_conf(5))

A GDI session is a new configuration object in Altair Grid Engine 8.2 which can be created, modified and deleted by managers or users that are members of the *sessionusers* access control list.

FORMAT

The sections below describe the format of the template file for session objects. Via the **-asi**, **-Asi**, **-msi**, **-Msi** options of the qconf(1) command session can be added and modified. The **-csi** switch can be used to create a session with default parameters. Any of these change operations can be rejected by the Altair Grid Engine system, as a result of a failed integrity verification.

qconf(1) **-ssil** will return a list of all existing sessions in an Altair Grid Engine system. Details of a session are shown with the command qconf(1) **-ssi**.

Note, Altair Grid Engine allows backslashes (“\”) be used to escape newline characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

The following list of parameters specifies the session configuration file content:

session_id

The session ID of a session. For sessions that should be created the value for this attribute has to be *NONE* so that the sge_qmaster(8) process can assign a new unique session ID.

owner

User name of the user that owns the session. If *NONE* is specified as username during the creation of a new session then the executing user of the configuration command will be the owner of that session.

Only managers and the session owner are allowed to modify or to delete an existing session and if a session gets created by root or a manager account on behalf of a regular user then that user should be a member of the *sessionusers* access control list.

duration

Duration of a session in a format as defined for *time* in sge_types(1) or the keyword *INFINITE*.

The *duration* influences the lifetime of a session. Lifetime of a session begins when the session is created and it ends when the session is not used for the specified amount of time defined by the *duration* attribute.

Lifetime of a session is automatically increased by adding *duration* to the *end_time* of that session when it is used. For sessions that have a duration of *INFINITE* the *end_time* will be *NONE* which means that the lifetime of the session will never end.

The default duration of a session is 900 seconds if this is not specified otherwise in the *qmaster_param* named *gdi_request_session_timeout* (see sge_conf(5))

The sge_qmaster(8) process tries to find sessions where the lifetime ended at least every 15 minutes and it will delete those sessions automatically. Although unused sessions will be deleted automatically it is recommended to delete sessions manually using the qconf(1) **-dsi** command once a session is not needed anymore.

start_time

Time when the session was created. *start_time* of a session cannot be specified. It is shown with qconf(1) **-ssi**.

end_time

Possible end time of a session or *NONE*.

After creation the *end_time* of a session is set to *start_time* plus *duration* for sessions defining a *time* as duration or it will be *NONE* if the duration is set to *INFINITE*.

end_time is moved forward when the session is used so that it still remains valid for at least the amount of time specified by duration after use.

If a non-*INFINITE* session was not used in its lifetime then it is tagged for deletion. The sge_qmaster(8) process tries to find tagged sessions at least every 15 minutes and it will delete those sessions automatically. Although unused sessions will be deleted automatically it is recommended to delete sessions manually using the qconf(1) **-dsi** command when a session is not needed anymore.

Sessions with an *INFINITE* duration are not deleted automatically.

The *end_time* of a session is shown by the commands qconf(1) **-ssi** and **-ssil**.

SEE ALSO

sge_intro(1), sge_types(1), qconf(1),

COPYRIGHT

sge_share_tree

NAME

share_tree - Altair Grid Engine share tree file format

DESCRIPTION

The share tree defines the long-term resource entitlements of users/projects and of a hierarchy of arbitrary groups thereof.

The current share tree can be displayed via the qconf(1) **-sstree** option. The output follows the share_tree format description. A share tree can be created and an existing one can be modified via the **-astree** and **-mstree** options to qconf(1). The **-sst** option shows a formatted share tree (tree view). Individual share tree nodes can be created, modified, deleted, or shown via the **-astnode**, **-dstnode**, **-mstnode**, and **-sstnode** options to qconf(1).

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

A tree is constructed from a root node, internal nodes and leaf nodes. The simplest useful tree consists of the root node and two leaf nodes. Internal nodes can be used to group leaf nodes. These internal nodes can either be abstract nodes with no relation to any Altair Grid Engine object, or nodes of type project, which relate to a defined Altair Grid Engine project (see sge_project(5)). sge_user(5)) or type project. See also the explanations in FORMAT and the EXAMPLES below.

FORMAT

The format of a share tree file is defined as follows:

- A new node starts with the attribute **id**, and equal sign and the numeric identification number of the node. Further attributes of that node follow until another **id**-keyword is encountered.
- The attribute **type** defines, if a sharetree node references a user (type=0), or a project (type=1)
- The attribute **childnodes** contains a comma separated list of child nodes to this node.
- The parameter **name** refers to an arbitrary name for the node or to a corresponding user (see sge_user(5)) or project (see sge_project(5)) if the node is a leaf node of the share tree. The name for the root node of the tree is “Root” by convention.
- The parameter **shares** defines the share of the node among the nodes with the same parent node.

- A user leaf node named ‘default’ can be defined as a descendant of a sge_project(5)) node in the share tree. The default node defines the number of shares for users who are running in the project, but who do not have a user node defined under the project. The default user node is a convenient way of specifying a single node for all users which should receive an equal share of the project resources. The default node may be specified by itself or with other sge_user(5)) nodes at the same level below a project. All users, whether explicitly specified as a user node or those which map to the ‘default’ user node must have a corresponding sge_user(5)) object defined in order to get shares. Do not configure a sge_user(5)) object named ‘default’.

EXAMPLES

Jobs of projects P1 and P2 get 50 shares, all other jobs get 10 shares.

```
id=0
name=Root
type=0
shares=1
childnodes=1,2,3
id=1
name=P1
type=1
shares=50
childnodes=NONE
id=2
name=P2
type=1
shares=50
childnodes=NONE
id=3
name=default
type=0
shares=10
childnodes=NONE
```

SEE ALSO

sge_intro(1), qconf(1), sge_project(5), sge_user(5).

COPYRIGHT

sge_user

NAME

user - Altair Grid Engine user entry file format

DESCRIPTION

A user entry is used to store ticket and usage information on a per user basis. Maintaining user entries for all users participating in a Altair Grid Engine system is required if Altair Grid Engine is operated under a user share tree policy.

If the **enforce_user** cluster configuration parameter is set to *auto*, a user object for the submitting user will be created automatically during job submission, if one does not already exist. The **auto_user_oticket**, **auto_user_fshare**, **auto_user_default_project**, and **auto_user_delete_time** cluster configuration parameters will be used as default attributes of the new user object.

A list of currently configured user entries can be displayed via the qconf(1) **-suserl** option. The contents of each enlisted user entry can be shown via the **-suser** switch. The output follows the *user* format description. New user entries can be created and existing can be modified via the **-auser**, **-muser** and **-duser** options to qconf(1).

Note, Altair Grid Engine allows backslashes (\) be used to escape newline (\newline) characters. The backslash and the newline are replaced with a space (" ") character before any interpretation.

FORMAT

A user entry contains four parameters:

name

The user name as defined for user_name in sge_types(1).

oticket

The amount of override tickets currently assigned to the user.

fshare

The current functional share of the user.

default_project

The default project of the user.

delete_time

Note: Deprecated, may be removed in future release.
The wall-clock time when this user will be deleted, expressed as the number of seconds elapsed since January 1, 1970. If set to zero, the affected user is a permanent user. If set to one, the user currently has active jobs. For additional information about automatically created users, see the **enforce_user** and **auto_user_delete_time** parameters in sge_conf(5).

SEE ALSO

sge_intro(1), sge_types(1), qconf(1), sge_conf(5).

COPYRIGHT

sgepasswd

NAME

sgepasswd - Altair Grid Engine password file format

DESCRIPTION

sgepasswd contains a list of user names and their corresponding encrypted windows passwords. If available, the password file will be used by sge_shepherd(8) and sge_execd(8). To change the content of this file please use the sgepasswd(1) command. It is not advised to change that file manually.

SEE ALSO

sge_intro(1), sgepasswd(1), Altair Grid Engine Installation and Administration Guide

COPYRIGHT

UGE_Starter_Service.exe

NAME

UGE_Starter_Service.exe - Altair Grid Engine execution daemon starter service

SYNOPSIS

UGE_Starter_Service.exe [-help | -install [ ] | -uninstall] | -add | -delete]

DESCRIPTION

On Windows (win-x86), UGE_Starter_Service.exe starts the sge_execd(8) at boot time with appropriate permissions.

OPTIONS

-help

Prints a listing of all options.

-version

Display version information for the *UGE_Starter_Service.exe* service.

-install [ ]

Installs the service. Only the local Administrator has the permissions to install the service. If name and option are not specified, the service is installed with “SYSTEM” credentials, which are not sufficient to start the sge_execd(8) properly. The local Administrators name and password can still be registered to the service by opening the Windows Services dialog and selecting the “Log on” tab.

-uninstall

Uninstalls the service. Only the local Administrator has the permissions to uninstall the service. Only a stopped service can be uninstalled. See the Windows command net.exe for information on how to stop a service.

-add

Adds an execution daemon to the list of daemons to start at boot time.

-delete

Deletes an execution daemon from the list of daemons to start at boot time.

SEE ALSO

sge_intro(1), sge_execd(8).

COPYRIGHT

sge_ca

NAME

util/sgeCA/sge_ca - Altair Grid Engine CSP Support control command

SYNTAX

sge_ca command [command options]

DESCRIPTION

sge_ca controls a simple Altair Grid Engine Certificate Authority that is used for the special Certificate Security Protocol (CSP) mode. CSP mode improves the security behavior of Altair Grid Engine by enabling OpenSSL secured communication channels and X509v3 certificates for authentication. In addition it is possible to export the key material or to create JKS keystores for the JMX connector. It follows a list of possible commands and command options to give an overview which functionality is available. For further details about every command refer to the COMMAND DETAILS section.

COMMAND OVERVIEW

sge_ca [-help]

show usage

sge_ca -init [command options]

create the infrastructure for a new Altair Grid Engine Certificate Authority with its corresponding files and directories and a set of keys and certificates for SGE daemon, root and admin user.

sge_ca -req | -verify <cert> | -sign | -copy [command options]

manipulate individual keys and certificates

sge_ca -print <cert> | -printkey | -printcrl <crl>

print out certificates, keys and certificate revocation lists in human readable form.

sge_ca -showCaTop | -showCaLocalTop [command options]

echo the $CATOP or $CALOCALTOP directory. This command is usually run as root on the qmaster host after a CA infrastructure has been created. If "**-cadir**" or "**-catop**" or "**-calocaltop**" are set the corresponding directories are printed.

sge_ca -usercert <user file> | -user <u:g:e> | -sdm_daemon <u:g:e> [command options]

are used for creation of certificates and keys for a bunch of users contained in <user file> or a single user <u:g:e>.

sge_ca -pkcs12 <user> | -sdm_pkcs12 <g> | -sys_pkcs12 [command options]

are used to export the certificate and key for user <user> or SDM daemon <g> in PKCS12 format and to export the SGE daemon certificate and key in PKCS12 format.

sge_ca -userks | -ks <user> | -sysks [command options]

are used for creation of keystore for all users with a certificate and key, the keystore for a single user <user> and the keystore containing the SGE daemon certificate and key.

sge_ca -renew <user> | -renew_ca | -renew_sys | -renew_sdm [command options]

are used to renew the corresponding certificates for user <user>, for the CA, for the SGE daemon certificate and for the SDM daemon <g> certificate.

where "**[command options]**" is a combination of the following options depending on the command. The COMMAND DETAILS section explains which options are usable for each command.

-days <days>

days of validity of the certificate

-md5

use MD5 instead of SHA256 as message digest

-sha1

use SHA-1 instead of SHA256 as message digest

-sha256

use SHA-256 as message digest which is the default when no md option -sha1, -sha-256, -md5 is given.

-encryptkey

use DES to encrypt the generated private key with a passphrase. The passphrase is requested when a key is created or used.

-outdir <dir>

write to directory <dir>

-cahost <host>

define CA hostname (CA master host)

-cadir <dir>

define $CALOCALTOP and $CATOP settings

-calocaltop <dir>

define $CALOCALTOP setting

-catop <dir>

define $CATOP setting

-kspwf <file>

define a keystore password file that contains a password that is used to encrypt the keystore and the keys contained therein

-ksout <file>

define output file to write the keystore to

-pkcs12pwf <file>

define a PKCS12 password file that contains a password that is used to encrypt the PKCS12 export file and the keys contained therein

-pkcs12dir <dir>

define the output directory

to write the exported PKCS12 format file to. Otherwise the current working directory is used.

COMMAND DETAILS

sge_ca -init [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <num days>]

The **-init** command creates a new Altair Grid Engine certificate authority and its corresponding files. Usually "**sge_ca -init**" is run by user root on the master host. If the options -adminuser, -cadir, -calocaltop, -catop and the Altair Grid Engine environment variables SGE_ROOT, SGE_CELL and SGE_QMASTER_PORT are set the CA directories are created in the following locations:
two letter country code, state, location, e.g city or your building-code, organization (e.g. your company name), organizational unit, e.g. your department, email address of the CA administrator (you!)

Certificates and keys are generated for the CA itself, for SGE daemon, for Altair Grid Engine install user (usually root) and finally for the Altair Grid Engine admin user.

How and where the certificates and keys are created can be influenced additionally by: **-days <days>** change the time of validity of the certificates to number of <days> instead of 365 days **-sha1** change the message digest algorithm from SHA-256 to SHA-1 **-md5** change the message digest algorithm from SHA-256 to MD5 **-encryptkey** encrypt the generated keys with a passphrase **-adminuser <user>** use as admin user **-cahost <host>** use as the CA master host **[-cadir <dir>] [-catop <dir> [-calocaltop <dir>]** set $CATOP and $CALOCALTOP to <dir> to use something different than the Altair Grid Engine default directories. Either -cadir <dir> has to be specified to replace $CATOP and $CALOCALTOP by the same directory or -catop <dir> for $CATOP and -calocaltop <dir> for $CALOCALTOP.

sge_ca -user <u:g:e> [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>][-adminuser <admin>] [-days <days>]

generate certificate and keys for <u:g:e> with u='Unix user account name', g='common name' and e='email address'. By default the certificate is valid for 365 days or by <days> specified with -days <days>. This command is usually run as user root on the qmaster host. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -sdm_daemon <u:g:e>

generate daemon certificate and keys for <u:g:e> with u='Unix user account name', g='common name' and e='email address'. By default the certificate is valid for 365 days or by <days> specified with "**-days <days>**". This command is usually run as user root on the qmaster host.

sge_ca -usercert <user file> [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>] [-encryptkey] [-sha1]

Usually sge_ca -usercert <user file> is run as user root on the master host. The argument <user file> contains a list of users in the following format:

eddy:Eddy Smith:eddy@griders.org
sarah:Sarah Miller:sarah@griders.org
leo:Leo Lion:leo@griders.org

where the fields separated by colon are: Unix user:Gecos field:email address

sge_ca -renew <user> [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>]

Renew the certificate for <user>. By default the certificate is extended for 365 days or by <days> specified with -days <days>. If the value is negative the certificate becomes invalid. This command is usually run as user root on the qmaster host. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -renew_ca [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>]

Renew the CA certificate. By default the certificate is extended for 365 days or by <days> specified with -days <days>. If the value is negative the certificate becomes invalid. This command is usually run as user root on the qmaster host. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -renew_sys [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>]

Renew the SGE daemon certificate. By default the certificate is extended for 365 days or by <days> specified with -days <days>. If the value is negative the certificate becomes invalid. This command is usually run as user root on the qmaster host. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -renew_sdm <g> [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>]

Renew the SDM daemon certificate of <g>, where <g> is the common name of the daemon. By default the certificate is extended for 365 days or by <days> specified with -days <days>. If the value is negative the certificate becomes invalid. This command is usually run as user root on the qmaster host. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -pkcs12 <user> [-pkcs12pwf <file>] [-pkcs12dir <dir>] [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>]

export certificate and key of user <user> 'the Unix user name' in PKCS12 format. This command is usually run as user root on the qmaster host. If -pkcs12pwf <file> is used the file and the corresponding key will be encrypted with the password in <file>. If -pkcs12dir <dir> is used the output file is written into <dir>/<user>.p12 instead of ./<user>.p12 . $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -sys_pkcs12 [-pkcs12pwf <file>] [-pkcs12dir <dir>] [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>]

export certificate and key of SGE daemon in PKCS12 format. This command is usually run as user root on the qmaster host. If -pkcs12pwf <file> is used the file and the corresponding key will be encrypted with the password in <file>. If -pkcs12dir <dir> is used the output file is written into <dir>/<user>.p12 instead of ./<user>.p12 . $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -sdm_pkcs12 <g> [-pkcs12pwf <file>] [-pkcs12dir <dir>] [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>]

export certificate and key of daemon <g> g='common name' in PKCS12 format. This command is usually run as user root on the qmaster host. If -pkcs12pwf <file> is used the file and the corresponding key will be encrypted with the password in <file>. If -pkcs12dir <dir> is used the output file is written into <dir>/<g>.p12 instead of ./<g>.p12 . $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -ks <user> [-ksout <file>] [-kspwf <file>] [-cadir <dir>][-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>]

create a keystore containing certificate and key of user <user> in JKS format where is the Unix user name. This command is usually run as user root on the qmaster host. If -kspwf <file> is used the keystore and the corresponding key will be encrypted with the password in <file>. The -ksout <file> option specifies the keystore file that is created. If the -ksout <file> option is missing the default location for the keystore is $CALOCALTOP/userkeys/<user>/keystore. This command is usually invoked by sge_ca

-userks. A prerequisite is a valid JAVA_HOME environment variable setting. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -userks [-kspwf <file>] [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>]

generate a keystore in JKS format for all users having a key and certificate. This command is usually run as user root on the qmaster host. If -kspwf <file> is used the keystore and the corresponding key will be encrypted with the password in <file>. The keystore files are created in $CALOCALTOP/userkeys/<user>/keystore. This command is run after user certificates and keys have been created with sge_ca -usercert <userfile> or if any of the certificates have been renewed. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -sysks [-kspwf <file>] [-cadir <dir>] [-catop <dir>][-calocaltop <dir>] [-adminuser <admin>]

generate a keystore containing the SGE daemon certificate and key in JKS format. This command is usually run as user root on the qmaster host. If -kspwf <file> is used the keystore and the corresponding key will be encrypted with the password in <file>. The keystore file is created in $CALOCALTOP/private/keystore. $CATOP and $CALOCALTOP maybe overruled by -cadir, -catop and -calocaltop.

sge_ca -print <cert>

Print a certificate where <cert> is the corresponding certificate in pem format.

sge_ca -printkey <key>

Print a key where <key> is the corresponding key in pem format.

sge_ca -printcrl <crl>

Print a certificate revocation list where <crl> is the corresponding certificate revocation list in pem format.

sge_ca -printcrl <crl>

Print a certificate revocation list where <crl> is the corresponding certificate revocation list in pem format.

sge_ca -req [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>] [-encryptkey] [-sha1] [-outdir <dir>]

create a private key and a certificate request for the calling user. This are created as newkey.pem and newreq.pem in the current working directory. If the option -outdir <dir> is specified in addition the files are created in <dir>.

sge_ca -sign [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>] [-adminuser <admin>] [-days <days>] [-encryptkey] [-sha1] [-outdir <dir>]

Sign a certificate request. The CA certificate under $CATOP (default: $SGE_ROOT/$SGE_CELL/common/sgeCA) and CA key from $CALOCALTOP (default: /var/sgaCA/{port$SGE_QMASTER_PORT|sge_qmaster}/$SGE_CELL) are used for the signature. If $CATOP and $CALOCALTOP are set to a different directory the information there is used. The certificate is created as newcert.pem in the current working directory or in <dir> if the option -outdir <dir> has been specified. In addition the option "**-days <number of days>**" can be specified to change the default validity from 365 to number of days.

sge_ca -verify <cert> [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>][-adminuser <admin>]

Verify a certificates validity where <cert> is the corresponding certificate in pem format. $CATOP and $CALOCALTOP can be overruled by -cadir, -catop and -calocaltop.

sge_ca -copy [-cadir <dir>] [-catop <dir>] [-calocaltop <dir>]

sge_ca -copy is run by a user to copy the users certificate and key on the master host to $HOME/.sge/port$SGE_QMASTER_PORT/$SGE_CELL/certs/cert.pem and the corresponding private key in $HOME/.sge/port$SGE_QMASTER_PORT/$SGE_CELL/private/key.pem which are used instead of the files in $CATOP and $CALOCALTOP. The command is only recommended for testing purposes or where $HOME is on a secure shared file system.

EXAMPLES

sge_ca -init -cadir /tmp -sha1 -encryptkey -days 31

create a CA infrastructure in /tmp with a certificate validity of 31 days using SHA-1 instead of SHA-256 as message digest.The keys are encrypted and a passphrase has to be entered during the creation of the different keys or during signing a certificate with the created CA key.

sge_ca -usercert /tmp/myusers.txt -cadir /tmp

/tmp/myusers.txt contains user1:My User:user1@myorg.org and user1 is a valid Unix user account. Create a key and certificate for user1.

sge_ca -userks -cadir /tmp

create a keystore for all users of the simple CA. The keystore is stored under /tmp/userkeys//keystore.

sge_ca -renew root -cadir /tmp -days -1

make the root certificate temporarily invalid.

sge_ca -renew_ca -days 365 -cadir /tmp

renew the CA certificate for 365 days

NOTES

Changes in the CSP hashing algorithms

All the SSL keys and certificates used by the CSP mode in SGE releases, prior to 8.6.x, are based on OpenSSL 1.0.0. Beginning with SGE releases 8.6.x, these certificates are based on OpenSSL 1.1.0, which breaks the backwards compatibility with the old encrypted keys and certificates. This is happening because OpenSSL now considers MD5 and SHA1 hashing algorithms as weak. Starting the SGE daemons, after an upgrade from a version prior to 8.6.x, with CSP enabled, to an SGE 8.6.x release, will fail, with the following observable message logged:

```
main|master01|E|commlib error: ssl error ([ID=336245134] in module "SSL routines": "ca md too
```

This message indicates that the MD5 hashing algorithm, used in keys and certificates generated using OpenSSL 1.0.0, is no longer compatible and supported with OpenSSL 1.1.0.

How can this be solved? If one finds himself in this situation, run the $SGE_ROOT/util/sgeCA/renew_all_certs.csh script (as root) to renew all the certificates in use, automatically It is recommended to run the $SGE_ROOT/util/sgeCA/renew_all_certs.csh script before starting the upgrade procedure, when upgrading from a SGE release version, prior to 8.6.X.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell.

RESTRICTIONS

sge_ca The command must be usually called with Altair Grid Engine root permissions on the master host. For more details on the permission requirements consult the detailed description for the different commands above.

FILES

sge_ca creates a file tree starting in **$CATOP** and **$CALOCALTOP**. The default for **$CATOP** is usually $SGE_ROOT/$SGE_CELL/common/sgeCA and for **$CALOCALTOP** /var/sgeCA/{port$SGE_QMASTER_PORT|sge_qmaster}/$SGE_CELL where the subpaths beginning with $ expands to the content of the corresponding environment variable.

In addition there may optionally exist the user certificate in $HOME/.sge/port$SGE_QMASTER_PORT/$SGE_CELL and the corresponding private key in $HOME/.sge/port$SGE_QMASTER_PORT/$SGE_CELL/private/key.pem which are used instead of the files in $CATOP and $CALOCALTOP. (see sge_ca -copy above)

SEE ALSO

sge_qmaster(8).

COPYRIGHT

sge_execd

NAME

sge_execd - Altair Grid Engine job execution agent

SYNOPSIS

sge_execd [-help]

DESCRIPTION

sge_execd controls the Altair Grid Engine queues local to the machine on which sge_execd is running and executes/controls the jobs sent from sge_qmaster(8) to be run on these queues.

OPTIONS

-help
Prints a listing of all options.

LOAD SENSORS

If a **load sensor** is configured for sge_execd via either the global host configuration or the execution-host-specific cluster configuration (See sge_conf(5).), the executable path of the load sensor is invoked by sge_execd on every configured **load_report_time** interval and delivers one or multiple load figures for the execution host (e.g. users currently logged in) or the complete cluster (e.g. free disk space on a network wide scratch file system). The load sensor may be a script or a binary executable. In either case its handling of the STDIN and STDOUT streams and its control flow must comply to the following rules:

The load sensor must be written as an infinite loop waiting at a certain point for input from STDIN. If the string “quit” is read from STDIN, the load sensor should exit. When an end-of-line is read from STDIN, a load data retrieval cycle should start. The load sensor then performs whatever operation is necessary to compute the desired load figures. At the end of the cycle the load sensor writes the result to stdout. The format is as follows:

- A load value report starts with a line containing only either the word “start” or the word “begin”.

- Individual load values are separated by newlines.
- Each load value report consists of three parts separated by colons (":") and containing no blanks.
- The first part of a load value information is either the name of the host for which load is reported or the special name "global".
- The second part is the symbolic name of the load value as defined in the host or global complex list (see complex(5) for details). If a load value is reported for which no entry in the host or global complex list exists, the reported load value is not used.
- The third part is the measured load value.
- A load value report ends with a line with only the word "end". After the word "end", it might be necessary to flush STDOUT to make the load values available for the execution daemon immediately.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell sge_execd uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which sge_qmaster(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SGE_EXECD_PORT

If set, specifies the tcp port on which sge_execd(8) is expected to listen for communication requests. Most installations will use a services map entry for the service “sge_execd” instead to define that port.

SGE_EXECD_KEEP_TRYING_TO_GET_CONFIG

If set to 1, keeps the sge_execd(8) from quitting if it can connect to the sge_qmaster(8) but does not get the configuration. This is the case e.g. if the host the sge_execd(8) is running on is not yet configured as an execution host at the sge_qmaster(8). If not set or if set to 0, the sge_execd(8) shows its normal behaviour, i.e. it quits if it does not get the configuration.

RESTRICTIONS

sge_execd usually is started from root on each machine in the Altair Grid Engine pool. If started by a normal user, a spool directory must be used to which the user has read/write access. In this case only jobs being submitted by that same user are handled correctly by the system.

FILES

sgepasswd contains a list of user names and their corresponding encrypted passwords. If available, the password file will be used by sge_execd. To change the contents of this file please use the sgepasswd command. It is not advised to change that file manually.

- /<cell>/common/configuration

Altair Grid Engine global configuration

- <sge_root>/<cell>/common/local_conf/<host>

Altair Grid Engine host specific configuration

- /<cell>/spool/<host>

Default execution host spool directory

- <sge_root>/<cell>/common/act_qmaster

Altair Grid Engine master host file

SEE ALSO

sge_intro(1), sge_conf(5), complex(5), sge_qmaster(8).

COPYRIGHT

sge_qmaster

NAME

sge_qmaster - Altair Grid Engine master control daemon

SYNOPSIS

sge_qmaster [-help]

DESCRIPTION

sge_qmaster controls the overall Altair Grid Engine behavior in a cluster.

OPTIONS

-help

Prints a listing of all options.

-version

Display version information for the *sge_qmaster* command.

ENVIRONMENTAL VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell *sge_qmaster* uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which *sge_qmaster*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SGE_EXECD_PORT

If set, specifies the tcp port on which *sge_execd*(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_execd" instead to define that port.

RESTRICTIONS

sge_qmaster is usually started from root on the master or shadow master machines of the cluster (refer to the *Altair Grid Engine Installation and Administration Guide* for more information about the configuration of shadow master hosts). If started by a normal user, a master spool directory must be used to which the user has read/write access. In this case only jobs being submitted by that same user are handled correctly by the system.

FILES

- <sge_root>/<cell>/common/configuration

Altair Grid Engine global configuration

- <sge_root>/<cell>/common/local_conf/<host>

Altair Grid Engine host specific configuration

- <sge_root>/<cell>/spool

Default master spool directory

SEE ALSO

sge_intro(1), sge_conf(5), sge_execd(8), sge_shadowd(8), Altair Grid Engine Installation and Administration Guide

COPYRIGHT

sge_shadowd

NAME

sge_shadowd - Altair Grid Engine shadow master daemon

SYNOPSIS

sge_shadowd

DESCRIPTION

sge_shadowd is a "light weight" process which can be run on so-called shadow master hosts in a Altair Grid Engine cluster to detect failure of the current Altair Grid Engine master daemon, sge_qmaster(8), and to start-up a new sge_qmaster(8) on the host on which the sge_shadowd runs. If multiple shadow daemons are active in a cluster, they run a protocol which ensures that only one of them will start-up a new master daemon.

The hosts suitable for being used as shadow master hosts must have shared root read/write access to the directory $SGE_ROOT/$SGE_CELL/common as well as to the master daemon spool directory (by default $SGE_ROOT/$SGE_CELL/spool/qmaster). The names of the shadow master hosts need to be contained in the file $SGE_ROOT/$SGE_CELL/common/shadow_masters.

RESTRICTIONS

sge_shadowd may only be started by root.

ENVIRONMENT VARIABLES

SGE_ROOT

Specifies the location of the Altair Grid Engine standard configuration files.

SGE_CELL

If set, specifies the default Altair Grid Engine cell. To address a Altair Grid Engine cell sge_shadowd uses (in the order of precedence):

The name of the cell specified in the environment variable SGE_CELL, if it is set.

The name of the default cell, i.e. **default**.

SGE_DEBUG_LEVEL

If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.

SGE_QMASTER_PORT

If set, specifies the tcp port on which sge_qmaster(8) is expected to listen for communication requests. Most installations will use a services map entry for the service "sge_qmaster" instead to define that port.

SGE_DELAY_TIME

This variable controls the interval in which sge_shadowd pauses if a takeover bid fails. This value is used only when there are multiple sge_shadowd instances and they are contending to be the master. The default is 600 seconds.

SGE_CHECK_INTERVAL

This variable controls the interval in which the sge_shadowd checks the heartbeat file (60 seconds by default).

SGE_GET_ACTIVE_INTERVAL

This variable controls the interval when a sge_shadowd instance tries to take over when the heartbeat file has not changed. The default is 240 seconds.

FILES

- <sge_root>/<cell>/common

Default configuration directory

- <sge_root>/<cell>/common/shadow_masters

Shadow master hostname file.

- <sge_root>/<cell>/spool/qmaster

Default master daemon spool directory

- <sge_root>/<cell>/spool/qmaster/heartbeat

The heartbeat file.

SEE ALSO

sge_intro(1), sge_conf(5), sge_qmaster(8), Altair Grid Engine Installation and Administration Guide.

COPYRIGHT

sge_shepherd

NAME

sge_shepherd - Altair Grid Engine single job controlling agent

SYNOPSIS

sge_shepherd

DESCRIPTION

sge_shepherd provides the parent process functionality for a single Altair Grid Engine job. The parent functionality is necessary on UNIX systems to retrieve resource usage information (see getrusage) after a job has finished. In addition, the sge_shepherd forwards signals to the job, such as the signals for suspension, enabling, termination and the Altair Grid Engine checkpointing signal (see sge_ckpt(1) for details).

The sge_shepherd receives information about the job to be started from the sge_execd(8). During the execution of the job it actually starts up to 5 child processes. First a prolog script is run if this feature is enabled by the **prolog** parameter in the cluster configuration. (See sge_conf(5).) Next a parallel environment startup procedure is run if the job is a parallel job. (See sge_pe(5) for more information.) After that, the job itself is run, followed by a parallel environment shutdown procedure for parallel jobs, and finally an epilog script if requested by the **epilog** parameter in the cluster configuration. The prolog and epilog scripts as well as the parallel environment startup and shutdown procedures are to be provided by the Altair Grid Engine administrator and are intended for site-specific actions to be taken before and after execution of the actual user job.

After the job has finished and the epilog script is processed, sge_shepherd retrieves resource usage statistics about the job, places them in a job specific subdirectory of the sge_execd(8) spool directory for reporting through sge_execd(8) and finishes.

sge_shepherd also places an exit status file in the spool directory. This exit status can be viewed with qacct -j JobId (see qacct(1)); it is not the exit status of sge_shepherd itself but of one of the methods executed by sge_shepherd. This exit status can have several meanings, depending on in which method an error occurred (if any). The possible methods are: prolog, parallel start, job, parallel stop, epilog, suspend, restart, terminate, clean, migrate, and checkpoint.

The following exit values are returned:

0	All methods: Operation was executed successfully.
99	Job script, prolog and epilog: When FORBID_RESCHEDULE is not set in the configuration (see sge_conf(5)), the job gets re-queued. Otherwise see “Other”.
100	Job script, prolog and epilog: When FORBID_APPERROR is not set in the configuration (see sge_conf(5)), the job gets re-queued. Otherwise see “Other”.
Other	Job script: This is the exit status of the job itself. No action is taken upon this exit status because the meaning of this exit status is not known.Prolog, epilog and parallel start: The queue is set to error state and the job is re-queued.Parallel stop: The queue is set to error state, but the job is not re-queued. It is assumed that the job itself ran successfully and only the clean up script failed.Suspend, restart, terminate, clean, and migrate: Always successful. Checkpoint: Success, except for kernel checkpointing: checkpoint was not successful, did not happen (but migration will happen by Altair Grid Engine).

RESTRICTIONS

sge_shepherd should not be invoked manually, but only by sge_execd(8).

FILES

sgepasswd contains a list of user names and their corresponding encrypted passwords. If available, the password file will be used by sge_shepherd. To change the contents of this file please use the sgepasswd command. It is not advised to change that file manually.

- <execd_spool>/job_dir/<job_id> job specific directory

SEE ALSO

sge_intro(1), sge_conf(5), sge_execd(8).

COPYRIGHT

uge_js_service.exe

NAME

uge_js_service.exe - Altair Grid Engine job starter service

SYNOPSIS

uge_js_service.exe [-help | -install | -uninstall]

DESCRIPTION

uge_js_service.exe cooperates with the sge_execd(8) to run jobs on Windows hosts. It is a Win32 service that is started at boot time under the "Local System" account, giving it the permissions to give any process the permissions to access the visible Desktop in order to let jobs display their GUI there.
This service runs independently of the sge_execd(8) (i.e. it is not started and stopped together with the execution daemon) and there can be only one uge_js_service.exe per host.

Jobs can request to be allowed to display their GUI on the visible desktop by requesting the resource "display_win_gui" (qsub -l display_win_gui=1).

The Windows administrator can operate this service using the usual Windows Service Control Manager or the "net" command. The "start" method starts the service if possible, the "stop" method stops it regardless of any jobs that might be still running. For a graceful shutdown, first run the "pause" method. This will start the shutdown procedure of the service and prevent it from accepting any further jobs. After this, the "resume" (aka "continue") method will fail as long as there are jobs running in the uge_js_service.exe. If no job is left in the service, the "resume" method will succeed and the service will stop.

OPTIONS

-help

Prints a listing of all options.

-version

Display version information for the *uge_js_service.exe* service.

-install

Installs the service. Only the local Administrator has the permissions to install the service.

-uninstall

Uninstalls the service. Only the local Administrator has the permissions to uninstall the service. Only a stopped service can be uninstalled. See the Windows command net.exe for information on how to stop a service.

SEE ALSO

sge_intro(1), sge_execd(8).

COPYRIGHT