



ALTAIR

ONLY FORWARD

Altair Allocator 2025.1.2

Administrator Guide

Contents

Altair Allocator Administrators Guide	3
Introduction.....	4
Start, Stop and Monitor Allocator.....	6
Sharing Licenses Across Multiple Sites.....	8
Configure Allocator.....	9
Allocator Procedures.....	13
Procedures Backward Compatibility.....	27
Allocator Usage Examples.....	29
Map Host Names.....	33
Resource Groups.....	35
Allocator and TIMEVAR.....	36
Passive Mode.....	37
License Overbooking.....	38
Overbooking in Allocator.....	40
Debugging.....	42
View Allocation Details.....	43
Web Interface.....	43
Standalone User Interface.....	47
Command Shell Output.....	48
Failover Server Candidates.....	49
Multiple Queues.....	53
Queues Example.....	54
PBS and Other Queues.....	58
Configure an LSF or OpenLava Queue.....	60
Configure an LSF or OpenLava elim.....	61
Legal Notices	64
Intellectual Property Rights Notice.....	65
Technical Support.....	71
Index	72

Altair Allocator Administrators Guide

1

Allocator works in conjunction with license monitoring and workload scheduling solutions such as Monitor and Accelerator.

This chapter covers the following:

- [Introduction](#) (p. 4)
- [Start, Stop and Monitor Allocator](#) (p. 6)
- [Sharing Licenses Across Multiple Sites](#) (p. 8)
- [Configure Allocator](#) (p. 9)
- [Allocator Procedures](#) (p. 13)
- [Procedures Backward Compatibility](#) (p. 27)
- [Allocator Usage Examples](#) (p. 29)
- [Map Host Names](#) (p. 33)
- [Resource Groups](#) (p. 35)
- [Allocator and TIMEVAR](#) (p. 36)
- [Passive Mode](#) (p. 37)
- [License Overbooking](#) (p. 38)
- [Overbooking in Allocator](#) (p. 40)
- [Debugging](#) (p. 42)
- [View Allocation Details](#) (p. 43)
- [Multiple Queues](#) (p. 53)
- [PBS and Other Queues](#) (p. 58)

This capability can be used to better utilize the software licenses by having the licenses shared among multiple queues.

Allocator reaches the objective of improving license utilization with the following rules:

- **Follow the demand:** When only one queue (or a subset of queues) requires a shared license, any leftover licenses are allocated to all queues based on weights.
- **Graduality:** The migration of licenses should be gradual so that the queue does not get more licenses than it is capable of using. For example, if a queue has access to 10 CPUs, it is clearly unnecessary to migrate more than 10 tokens of a license to that queue.
- **Share according to policies:** If multiple queues require a shared license, the license should be distributed between the queues based on management policies.

Introduction

Allocator works in conjunction with license monitoring and workload scheduling solutions such as Monitor and Accelerator.

It keeps track of total number of licenses in the pool and those in use, and it allocates available licenses across geographic sites and business units via comprehensive resource management policies.

Another facet of sharing software resources across multiple sites is having clear visibility into how licenses are being utilized across an organization, along with the ability to prioritize and allocate licenses based on organizational needs.

Allocator gives organizations the software license policy management features they need to adjust license allocation and maximize the utilization of their software assets. Allocator's advanced features provide the most comprehensive license management and allocation solution available:

- Allocator's metrics provide visibility into license utilization history and status across multiple sites
- Overbooking allows greater utilization of licenses
- Allocator's advanced matching algorithm, including matching by process ID (PID), allows for more accurate allocation of licenses across sites
- Organizations can set internal license management policies by setting a maximum number of licenses that can be allocated to any given site

Allocator with Accelerator

Allocator greatly enhances Accelerator with these features:

- Supports LSF clusters.
- Provides new set of graphical historical reports that show how Allocator allocated licenses across the sites.
- Operates with its own vovserver.

Because of its descendancy from MultiQueue, there are several instances of "MQ" in the name of the commands, procedures, directories. Altair Allocator is the name of the new product; MultiQueue is the name of the previous daemon that was attached to an Accelerator instance.

Licensing of Allocator

The `la` project requires the following licenses:

- One token of the license feature "server_la"
- Multiple tokens of the license feature "jobs_la", as determined by the total number of running jobs detected by the system

Should a license violation occur, an alert will be issued. Currently, all functionality will continue to be in place; this policy may change.

Common Commands in Allocator

The following commands are available in Allocator:

- `lamgr`: The tool to start and stop Allocator

- `vovshow -la`
- `vovshow -la all`
- `vovshow -la inuse`
- `vovshow -la alloc`
- `vovshow -la queued`
- `vovshow -lares <NAME_OF_RESOURCE>`
- `vovla_gui -r <NAME_OF_RESOURCE>`

The following system commands are new and called only by Allocator.



Note: These commands are not intended to be used by administrators or users; they are to be called only by Allocator.


- `vovlalm`: Probe a Monitor instance
- `vovlanc`: Probe an Accelerator instance
- `vovlalsf`: Probe LSF

Start, Stop and Monitor Allocator

1. Pick a host as the main controller for the sites. It is recommended to select a host that runs an Accelerator queue.

```
% lamgr start
```

This starts the vovserver for Allocator and the auxiliary daemon which is called `vovlad`, which is mostly used to track changes in the `config.tcl` file.

 **Tip:** Use the command `lamgr -h` for more information on the various options available with the `lamgr` command.

2. To manually start `vovlad`, follow this procedure:

```
% vovproject enable la
% cd      `vovserverdir -p vovlad`
% vi config.tcl
% vovlad -v
```

3. To start and stop `vovlad` in the background, use the following utility `vovdaemonmgr`:

```
% vovproject enable la
% vovdaemonmgr stop vovlad
% vovdaemonmgr start vovlad
```

4. Display the usage message for `vovlad` with the option `-h`:

```
vovlad: Usage Message
```

DESCRIPTION:

The main Allocator daemon used as part of the Allocator product to allocate licenses across multiple Accelerator sites and multiple LSF clusters.

USAGE:

```
% vovlad [OPTIONS] &
```

OPTIONS:

```
-config <file>      -- Specify a configuration file.
                     (default is config.tcl)
-h                  -- Help usage message.
-loopSleep <t>      -- Set sleep time for each loop.
                     (default 10s)
-q                  -- Quiet mode: inverse of -v.
-rank <N>           -- Ignored: use SetRank instead.
-v                  -- Verbose flag (may be repeated).
```

EXAMPLES:

```
% vovlad >& vovlad.log &
% vovlad -v
% vovlad -v -v -v
```

5. To terminate the vovserver for Allocator, use

```
% lamgr stop
```

6. You can check the operation of Allocator through the web browser interface Allocator page. You can also use the command line as shown below:

```
% vovshow -la  
...
```

See Also

[la mgr](#)

Sharing Licenses Across Multiple Sites

Summary information for vovlad


Working directory	<code>la.swd/vovlad</code>
Config file	<code>la.swd/vovlad/config.tcl</code>
Info file	<code>la.swd/vovlad/info.tcl</code>
Template config	<code>\$VOVDIR/etc/config/vovlad/config.tcl</code>

Configure Allocator

To configure Allocator, you need to create a file called config.tcl. This is the only file required for configuring Allocator.

This file consists of the following parts:

- Declaration of shared resources
- Declaration of sites
- Optional declaration of user maps
- Optional declaration of local names for resources
- Optional declaration of the Allocator systems to use
- Optional declaration of constraints like stand-by allocations and out-of-queue reservations

 **Note:** The declaration of sites and resources must precede configuration of sites or resources. If an attempt is made to configure a site or resource, such as setting the weight of a resource for a given site, before the site or resource has been declared, Allocator will generate an alert.

The config.tcl file can be built starting from a template found in \$VOVDIR/etc/config/vovlad/config.tcl as shown in the following example.

```
### LicenseAllocator(tm) configuration file.
###
### This file should be placed in the la.swd/vovlad directory.
###
### Examples of a few basic commands are included in this file. Please uncomment
### relevant lines and modify as appropriate for your installation.

### Control the rank of LicenseAllocator. Default is 30. It must be >= 20 and <
1000.
### This allows the setup of a failover LicenseAllocator.
### The main LicenseAllocator should have a rank higher than the failover.
LA::SetRank 30

#####
### DECLARATION SECTION ###
#####
# Declare LicenseMonitor, Resources, and Sites in this section.
# All these declaration must precede configuration in the Configuration section.

### Specify server and port for LicenseMonitor. Alternately, you can let LA
### automatically detect the host and port from the registry, by using the keyword
### "auto" in place of host:port specification, and using LM project name (licmon
### by default)
### If the LicenseMonitor instance has SSL enabled, you must specify the -ssl
### flag on this command.
# LA::AddLicenseMonitor "lmsrv:5555" -version 2016.09 -ssl 1
# LA::AddLicenseMonitor "lmsrv1:5555 lmsrv2:6666"
# LA::AddLicenseMonitor "auto" -ssl 1 -version SAME

### Add a resource to share across all sites.
### LA::AddResource <name> <tag/feature | qty> <optional map>

# LA::AddResource M5 10 ""
```

```
# LA::AddResource Lic:dc FlexLM/SYNOPSYS/Design-Compiler ""
# LA::AddResource WAN:lic_drc FlexLM/EDA/lic_drc ""

### Alternatively, you can add all features we find in LicenseMonitor.
# LA::AddAllResources -includeTag _wan -useGroups 0 -resmapType License

###
### Define resource groups, i.e. groups of resources that
### should be handled together.
###
# LA::DefineResourceGroup "WAN:Spice" {
#   LA::AddResource "India:spice" FLEXLM/IN/spice ""
#   LA::AddResource "Russia:spice" FLEXLM/RU/spice ""
# }

### Add two or more sites. LicenseAllocator is not needed for only one site.

### Syntax:
### AddSite <NC instance> <nickname> { <resources> } [OPTIONS]
### where options can be:
### -port <NC web port number>
### -version <RTDA version>
### -remotehost <host>
### -defaultweight <WEIGHT>
### -ssl 1
### -autodetect <Automatically detect the host and port from the
### registry, for the specified NC instance>

### The <nickname> should be a short name for the site. We recommend using 2 to 4
### characters,
### and no spaces.

### The <resources> are a list of alternating resource names
### and weights. When there is contention, each site's allocation
### is the ratio of its weight divided by the sum of the weights
### for all sites where that resource is in demand.
### For the example below, when San Jose and Shanghai both have
### jobs needing License:h1, San Jose gets 20/(20+40) = 1/3
### and Shanghai gets 40/(20+40) = 2/3
### BEST USAGE: it is best to leave the resources list empty and control the
### weights of resources with LA::SetResourceWeight and with -defaultweight in
### AddSite.
###
### If the NC instance has SSL enabled, this command must include -ssl 1 flag to
### enable
### communication with that NC site. When using SSL, make sure you specify the Web
### Port
### for the NC instance, and not the VOV port.

###
### If a remote host is specified, the information from the site is gathered
### by doing an ssh to that host and by compressing the data with gzip.
### You must set up ssh so that no password is requested.
# LA::AddSite vncju@jupiter "SJ" {
# } -port 6271 -version 2016.09 -remotehost sj-login -ssl 1

# LA::AddSite vncpl@pluto "CN" {
# } -port 6271 -version 2016.03

# LA::AddSite vncsn "SN" {
# } -autodetect -version 2016.09 -defaultweight 240 -ssl 0

##### ADD AN LSF CLUSTER
```

```

### You need to use the options -lsf and setup a few directories and scripts.
### Please refer to documentation for details.

# LA::AddSite lsf1@uni00 lsfUni {} \
# -lsf -user cadmgr -mqdir [vtk_path_expand ./lsf_mq_dir] \
# -remotehost uni00 -timezone PST8PDT

#####
### CONFIGURATION SECTION ###
#####
# Configure various aspects of LicenseAllocator. Note, all LicenseMonitor,
# Resources, Sites must have already been declared above in the Declaration
# section.

### Define different names for resources for specific sites
### LA::DefineRemoteResourceName <site> <resourcemap> <remotename>
# LA::DefineRemoteResourceName vnc@pluto License:h1 lic_h1

### Define a minimum Stand-By quantity for a resource in a site.
# LA::SetMinQuantity CN License:h1 3

### Allow for reservations for out-of-queue jobs
### Reserve 3 licenses of License:abc for user edward to use out-of-queue.
# LA::SetReserveForUser License:abc edward 3
### Reserve 10 license of License:abc for any out-of-queue user.
# LA::SetReserveForUser License:abc * 10

### Reload historical metric data for all sites and resources for the last 30 days
# LA::ReloadHistoricalMetrics 30d

##### ADVANCED SETTINGS #####

### MQdebug array. If you need to generate additional debugging output for a
specific resource
### set the variable MQdebug($RESOURCE) to 1. Reset it to 0 to
stop the output.
###
global MQdebug
# set MQdebug(License:abc) 1

### MQ(loopSleep) The delay between cycles.
###
set MQ(loopSleep) 30

### MQ(useAllocFiles) Used only for LSF sites. If set, create files lsfddata/
$NICNAME/elim.sitealloc_{status,total}
### Default is 0.
set MQ(useAllocFiles) 0

### If you want to execute a script at every cycle
### This can be repeated.
#
# LA::RegisterScript $tclScript
#

```

Modify the Configuration File

The configuration file can be changed at any time, even as Allocator is running. The changes will be automatically detected within 1 minute or less.

The changes may include:

- Adding or removing resources
- Adding or removing queues
- Changing the distribution weights of resources among the queues

Configuration for FairShare with Accelerator

Allocator can be configured to compute FairShare for the Accelerator sites that it manages. In the default state, Allocator does not compute FairShare for the NC sites.

FairShare is computed on a global level, taking into account jobs from all the Accelerator sites configured for external FairShare computation. In other words, jobs from all the Accelerator sites configured for FairShare computation are pooled together to compute FairShare and the resulting FairShare values are pushed to these Accelerator sites.

If an Accelerator site is configured for external FairShare computation, the **FS_SERVER_URL** property on the Accelerator server will be set to point to Allocator. Also, its **fairshare.cgi** page and the output of `vovfsgroup show` will have links to Allocator's FairShare page.

Allocator Procedures

In this section, the procedures that may be used in the Allocator configuration file are described.

LA::AddLicenseMonitor

Description

Use this procedure to declare which Monitor system Allocator talks to. This procedure may be called multiple times if there are multiple Monitor systems that need to be connected to Allocator.



Note: Although `-nickname` is an optional parameter if only a single Monitor is declared using this command, it must be specified with a unique nickname if multiple Monitor are declared through multiple `AddLicenseMonitor` commands.



Note: Current version of Allocator only supports connection to one Monitor instance. Support for multiple Monitor instances will be added in a future release.

Usage

LA::AddLicenseMonitor <host:port> [OPTIONS]

Required Arguments

<host:port>	The first argument is a list of host:port elements (normally a list of just one).
[host2:port2] ...	Alternately, the keyword "auto" can be used in place of the host:port specification, to detect the host and port automatically from the registry for the Monitor project name specified using <code>-project</code> .

Options

-project PROJECTNAME	Specify the name of the Monitor project. The default value is "licmon".
-access [normal vtk]	This option is deprecated. The only available access is now vtk.
-local-file FILENAME	This is used only for debugging. Please do not use.
-nickname NICKNAME	Specify a short nickname to refer to this Monitor instance.



Note: If the `config.tcl` file contains multiple `AddLicenseMonitor` commands, each Monitor declaration MUST include a unique nickname.

`-ssl [0|1]`

Specify whether to use SSL to connect to this Monitor instance or not. This depends on whether this Monitor instance is running with SSL enabled or is running with SSL disabled.

Examples

```
LA::AddLicenseMonitor bear:5555
LA::AddLicenseMonitor "bear:5555 calicmon:5555"
LA::AddLicenseMonitor bnglic:5555 -nickname BL
LA::AddLicenseMonitor lnx123:7755 -project licmon_us -
ssl 1
```

LA::AddResource

Description

Declare a shared resource.

Usage

`LA::AddResource <RESOURCE_NAME> <TYPE> <MAP> [OPTIONS]`

Required Arguments

<code><RESOURCE_NAME></code>	Name of the resource to be shared.
<code><TYPE></code>	Either an integer representing the total number of units to be shared, or string of the form "FlexLM/tag/feature", where <i>tag/feature</i> is the name of a FlexNet Publisher feature, inclusive of tag. These names are the ones used by Monitor. The "FlexLM" part of the type is case-insensitive.
<code><MAP></code>	A resource expression to which the resources is to be mapped.

Options

<code>-expr <EXPRESSION></code>	EXPRESSION is an integer expression where the token '\$N' represents the number total number of tokens as determined by TYPE. An example of this option is shown below.
---------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

-slowdown <RATIO> This number is used to slow down the transfer of tokens to a queue that needs them. Specifically, if A is the number of allocated tokens now and T is the target number of tokens, and R is the "slow down ratio" set by this option, then the actual number of tokens transferred to the queue in each cycle of Allocator is $\text{ceil}((T-A) * R)$. The default value of R is 0.5 while the valid range is from 0.1 to 1.0. This option is not supported in this release.

-tags <LIST_OF_TAGS> LIST_OF_TAGS is a list of the tags that have licenses managed by Allocator. This option is necessary if you have a mixture of wannable and non-wannable licenses.

-debug <LEVEL> Enable or disable debugging on this resource and set the debug level. A level of 0 disables debugging, and any positive integer enables it.

-overbook [0|1] Enable or disable overbooking on this resource. A level of 0 disables overbooking, and 1 enables it.



Note: Enabling overbooking will automatically disable handle/job matching.

-match [0|1] Enable or disable handle/job matching on this resource. A level of 0 disables matching, and 1 enables it.



Note: Matching can only be enabled if overbooking is disabled. Enabling overbooking will automatically disable matching.

Examples

```
# 10 tokens to be shared.
LA::AddResource MyTokens 10 ""
# Wan:licx is derived from the FlexNet Publisher
feature called 'licx'
```

```
LA::AddResource Wan:licx FLEXLM/licx "" -overbook 1;
```

LA::DefineResourceGroup

Description

If you have the same feature serviced by different FlexNet Publisher daemons, it may be advantageous to use this command to keep the various sources separated and allows Allocator to manage them as a group. This command is also recommended if you are using VOV_LM_VARNAMEs to automatically set the value of the LM_LICENSE_FILE variable.

Usage:

```
LA::DefineResourceGroup <GROUP_NAME>  
<LIST_OF_ADD_RESOURCE_CALLS>
```

Required Arguments

<GROUP_NAME>	Name of the resource group.
<SCRIPTS THAT CALLS LA::AddResource>	Normally the script is a simple sequence of LA::AddResource calls (see examples below).

Examples

In this example, assume there are 3 sources for the license "hsim": US, CA, EU. Each source provides 10 tokens, for a total of 30 tokens. If you simply tell Allocator to manage the 3 sources, and if there is contention for the resource, then Allocator will try to balance each of the 3 sources.

```
# Wrong way to deal with multiple sources  
LA::AddResource WAN_US:hsim FLEXLM/US/hsim ""  
LA::AddResource WAN_CA:hsim FLEXLM/CA/hsim ""  
LA::AddResource WAN_EU:hsim FLEXLM/EU/hsim ""
```

Instead, if you use a "resource group", then Allocator will try to balance the total number of tokens for the three sources, thus allowing each individual source to be allocated independently of the other two.

```
# Correct way to deal with multiple sources  
LA::DefineResourceGroup WAN:hsim {  
    LA::AddResource WAN_US:hsim FLEXLM/US/hsim ""  
    LA::AddResource WAN_CA:hsim FLEXLM/CA/hsim ""  
    LA::AddResource WAN_EU:hsim FLEXLM/EU/hsim ""  
}
```

LA::AddAllResources

Description

If you have many WANable resources, it may be convenient to declare them all at once with this command. This procedure queries all Monitor systems for all features in selected tags and

declares all Monitor systems as resources to be managed by Allocator.

Usage

LA::AddAllResources [OPTIONS]


Options

-resmapType <TYPE> Specify the type to be used for the resmaps. For example, if the resmap type is "XXX" then the Allocator resources will be called XXX:featureName. The default value for -resmapType is "LicenseWAN".

-useGroups <BOOLEAN> If -useGroup is true, then the features in different tags are grouped together with LA::DefineResourceGroup otherwise the available tokens are all aggregated together and the resource is managed as one.

-excludeTag <RX> Used to exclude tags that are not supposed to be managed by Allocator. The option may be repeated.

-includeTag <RX> Used to include tags that are supposed to be managed by Allocator. The option may be repeated.

 **Important:** The -includeTag option is processed before the -excludeTag option. If the option is not specified, all tags are included, else only the mentioned tags are included. If a tag is both "included" and "excluded", then the exclusion rule prevails.

-orderTags <TAGLIST> Define a partial order of tags. The option may be repeated.

Examples

```
# Exclude a specific tag called SNPS. Remember that
all strings are Regular expressions
LA:AddAllResources -excludeTag {^SNPS$}

# Only add resources that have tag matching "_wan";
```

```
# All remaining resources will have type LicenseWAN

LA:AddAllResources -includeTag _wan -resmapType
  "LicenseWAN"

# Only add resources that have tag matching "_wan",
# but use LA::DefineResourceGroups for a finer
# management of the
# resources.
LA::AddAllResources -useGroups 1 -includeTag _wan -
resmapType "LicenseGRP"
```

LA::AddSite

Description:

Define a queue, such as an Accelerator site, managed by Allocator.

Usage:

```
LA::AddSite <QUEUE@HOST> <NICKNAME> [{LIST_OF_WEIGHTS}]
[OPTIONS]
```

Required Arguments

<SITE@HOST>	Specify the site@host name. This is easy for an Accelerator site. For LSF, you may want to pick an arbitrary name for SITE and choose the master host for HOST. Alternately, the @host may be omitted if using the -autodetect option.
<NICKNAME>	Specify a site nickname. This is a short name used in the browser interface and also used in other API calls. Make it short with no spaces.

Options

{LIST_OF_WEIGHTS}	An optional Tcl list of resource, weight pairs. The weights are used to decide the allocation of the resources in case of contention. We currently recommend to leave this empty and to use LA::SetResourceWeight instead.
-la [0 1]	Used by the parent Allocator in a hierarchical Allocator setup to define a child LA (using the -la 1 option).

Examples:

```
LA::AddSite child_la@jupiter
  CHILDLA {} -port 8787 -la 1 -
version SAME -defaultweight 200
```

-defaultweight <WEIGHT>	Specify the default weight for this site. This weight is used if no weight is specified for this site on any resource.
-port <PORT>	Specify the port for this site. Note, when using SSL, it is essential to specify the web port of the NC instance, and not the VOV port. This option may be omitted ONLY if using -autodetect.
-access <ACCESS_METHOD>	This option is deprecated. The only available access is now vtk.
-ssl <BOOLEAN>	Specify whether this site is SSL enabled or not.
-version <VERSIONID>	Specify the VOV software version of the remote site. This allows the use of different versions at different sites. The default value is 'SAME' to mean the same version used by Allocator.
-lsf	Declare this to be a LSF site. For this type of site, other options shown below apply.
-lsfdur BOOLEAN	Only for LSF sites. This option controls whether to honor the <code>duration=N</code> statement in <code>rusage[]</code> . The default value is 0, which ignores the duration statement.
-user USER	Only for LSF sites. This is the name of the user that runs bjobs on the remote host where LSF is running. The 'user' is used in a 'ssh user@host ...' command.
-lsfdir DIR	Only for LSF sites. This is the directory on the remote LSF host in which to run the bjobs command. This option is explained in detail in PBS and Other Queues .
-local-bjobs-file FILE	Only for LSF sites. Only for debugging. Please do not use.
-timezone TZ	Only for LSF sites. The time zones in which the LSF master is running.
-autodetect	Automatically detect the host and port from the registry, for the specified

Accelerator site. This option has no effect for LSF or PBS.

-fairshare

Instruct Allocator to compute the FairShare for the Accelerator site. This option is all that is needed to compute FairShare in Allocator and push it to the Accelerator site. It also disables the internal FairShare computation in the Accelerator site.

If it has not received FairShare updates from Allocator, an Accelerator site configured for external FairShare computation will compute FairShare internally. The default wait interval is 120 seconds. This interval can be configured on the Accelerator instance using the `fairshare.nocompute.maxUpdateInterval` trace parameter. The minimum valid value is 60 seconds, and the maximum valid value is 3600 seconds.

Examples

```
LA::AddSite vnc@jupiter      US {} -version 2024.1.1 -  
port 7788  
LA::AddSite vnc_ca@calnx03 CA {} -version SAME      -  
port 6227 -defaultweight 200  
LA::AddSite vncx2@--        IN {} -autodetect -version  
SAME -defaultweight 200  
LA::AddSite lsv1@ux01        NY {} -defaultweight 330  
-scheduler LSF -lsfdur 1 -lsfdir ./lsf_la_dir -  
timezone EST5EDT  
LA::AddSite pbs1host        PBS1 {} -defaultweight 330 -  
scheduler PBS
```

LA::DefineUserMap

Description

For some machines, assume that a specified user is using the licenses. Used on installation that include both Windows and UNIX taskers in which the Windows hosts may be running as a different user from the one that submitted the job.

Usage

```
LA::DefineUserMap <HOST_PATTERN> <USER>
```

Required Arguments

<PATTERN> Hostname glob pattern.

<USER>

User name to be used for all information referring to the hosts matching the pattern.

Examples

```
# The actual user of licenses on machines that match
the string 'farm1' is actually sysmgr.
LA::DefineUserMap farm1      sysmgr

# The user of licenses on machines that match 'farm[a-
z]' is actually bspice.
LA::DefineUserMap farm[a-z]  bspice

# The users on machines called like winJ.* are mapped
to user "john".
LA::DefineUserMap winJ.*  john
```

LA::DefineRemoteResourceName

Description

By default, the name of the resource map on each site is the name declared in the `LA::AddResource` statement. Sometimes, however, a resource map should have different names on different sites.

Usage

```
LA::DefineRemoteResourceName <SITE@HOST>
<SHARED_RESOURCE> <REMOTENAME>
```

Required Arguments

<SITE@HOST>	Site name or nickname.
<SHARED_RESOURCE>	Shared resource name
<REMOTENAME>	Remote resource name.

Examples

```
# On the site vnc@comet the name to be used for
resource
# License:spice is old_spice.
LA::DefineRemoteResourceName vnc@comet      License:spice
old_spice
LA::DefineRemoteResourceName vnc@ny         License:spice
license_spice
```

LA::MarkVqAsOoq

Description

Vendor-queue (VQ) tokens are marked as out-of-queue (OOQ) by default. Use this proc to change this behaviour for all the resources managed by Allocator. This proc, if called, must be called before any calls to `LA::MarkResourceVqAsOoq`

Usage

```
LA::MarkVqAsOoq <isVqAsOoq>
```

Required Arguments

<isVqAsOoq>

Set this to 0 (default is 1) to disable marking vendor-queued (VQ) tokens as out-of-queue (OOQ) for all resources.

Examples

```
preBlock {  
  LA::MarkVqAsOoq 0  
}
```

LA::MarkResourceVqAsOoq

Description

Use this proc to control whether Allocator marks vendor-queue (VQ) tokens as out-of-queue (OOQ) for the given resource. The default behaviour is to mark VQ tokens as OOQ. This overrides the global setting set by `LA::MarkVqAsOoq` and all calls to this proc must be after any call to `LA::MarkVqAsOoq`.

Usage

`LA::MarkResourceVqAsOoq <Resource> <isVqAsOoq>`

Required Arguments

<Resource>

Name of a resource.

<isVqAsOoq>

Set this to 0 (default is 1) to disable marking vendor-queued (VQ) tokens as out-of-queue (OOQ) for all resources.

Examples

```
preBlock {  
  LA::MarkResourceVqAsOoq License:sqlite 0  
}
```

LA::SetMapForResourceInSite

Description

This procedure allows using different maps on each Accelerator site for each resource managed by Allocator. This only applies to Accelerator sites, not to LSF sites. This procedure provides fine control on the resource map of a resource in a site. This may be used to change the order of components in resource groups. To delete the map, use the value "-" for MAP.

Required Arguments

<SITE>

Site name or nickname.

<RESOURCE>

Name of a resource

<MAP>

Map to be used.

Examples

```
# The users on machines called like winJ.* are mapped  
to user "john"
```

```
# US before EU licenses.
LA::SetMapForResourceInSite SanJose License:ncsim
"License:US_ncsim OR License:EU_ncsim"
# EU before US licenses.
LA::SetMapForResourceInSite Munich License:ncsim
"License:EU_ncsim OR License:US_ncsim"
# Use default value.
LA::SetMapForResourceInSite Pudong License:ncsim -
```

LA::SetMinQuantity

Description

Define a minimum amount of a resource to assign to a site even if the demand for that site is zero, that is, the site is in stand-by with respect to that resource.

Required Arguments

<SITE>	Site name or nickname.
<RESOURCE>	Name of a resource
<N>	Minimum number of tokens assigned to the site even if there is no demand.

Examples

```
### Allocate at least 3 tokens of License:abc to site
with nickname NY
LA::SetMinQuantity NY License:abc 3
```

Comments

If unset, Allocator will need to be restarted. To avoid restarting, change the min/max values as follows:

2016.09 releases:

```
LA::SetMinQuantity <site> <resource> -1
LA::SetMaxQuantity <site> <resource> -1
```

2017 releases:

```
LA::SetMinQuantity <site> <resource> NO_MIN
LA::SetMaxQuantity <site> <resource> NO_MAX
```

LA::SetMaxQuantity

Description

Define the maximum amount of a resource to assign to a site even if the demand for that site is high.

Required Arguments

<SITE>	Site name or nickname.
<RESOURCE>	Name of a resource

<N> Maximum number of tokens assigned to the site even if there is no demand.

Examples

```
### Allocate at most 25 tokens of License:abc to site  
with nickname NY  
LA::SetMaxQuantity NY License:abc 25
```

LA::SetReserveForUser

Description

Reserve tokens for out-of-queue users. You can reserve tokens for any named user or for anybody by using the string "*" as user name.

Required Arguments

<RESOURCE> Name of a resource to be reserved.

<USERSPEC> The user for whom the resource is reserved. Could be * for 'any user'

<TOKENS> Number of tokens to be reserved.

Examples

```
#  
# Allow 8 out-of-queue tokens for user edward and 10  
# overall.  
#  
LA::SetReserveForUser License:abc edward 8  
LA::SetReserveForUser License:abc * 10
```

LA::SetResourceWeight

Description

Set the weight of a resource for a site.

Required Arguments

<SITE> Site name or nickname.

<RESOURCE> Name of a resource to set the weight on.

<WEIGHT> Specifies the weight for the resource for this site. It can either be an integer or the string "DO_NOT_SHARE" or "FROM_SITE". With "DO_NOT_SHARE", Allocator will never assign tokens to the specified site. If the weight is specified as "FROM_SITE" the default weight for the site will be used.

Examples

```
# LA::SetResourceWeight SITE RESOURCE WEIGHT  
LA::SetResourceWeight A License:a 200
```



```
LA::SetResourceWeight B License:x DO_NOT_SHARE
```

LA::GetResourceField

Description

Get the value of a specific field from the resource.

Required Arguments

<SITE>	Site name or nickname.
<RESOURCE>	Name of a resource
<FIELD>	The field whose value needs to be obtained. The list of fields that can be queried through this procedure can be obtained by running the following command:

```
vovselect FIELDNAME from  
LAALLOCATIONS
```

<DEFAULT>	Default value to be returned if the value of this field cannot be obtained.
-----------	-----------------------------------------------------------------------------

Examples

```
LA::GetResourceField A License:a WEIGHT  
LA::GetResourceField B License:x PRIORITY
```

LA::GetSiteField

Description

Get the value of a specific field from the site.

Required Arguments

<SITE>	Site name or nickname.
<FIELDS>	The fields whose value needs to be obtained (comma separated). The list of fields that can be queried through this procedure can be obtained by running the following command:

```
vovselect FIELDNAME from LASITES
```

<DEFAULT>	Default value to be returned if the value of this field cannot be obtained.
-----------	-----------------------------------------------------------------------------

Examples

```
LA::GetSiteField A SSLENABLED  
LA::GetSiteField B VERSION
```

LA::SetMinExpirationTime

Description

Get the value of a specific field from the site.

Required Arguments

~~<MIN_EXPIRATION_TIME>~~ Minimum expiration time for the resources managed to Allocator.

Examples

```
LA::SetMinExpirationTime 5m
```

LA::SetRank

Description

Set the rank of this Allocator instance. If resources are managed by multiple Altair Accelerator products such as Allocator or Accelerator, the highest ranked product wins.

Required Arguments

<RANK> The rank for this Allocator instance. Rank must be between 20 and 1000.

Examples

```
LA::SetRank 35
```


LA::ReloadHistoricalMetrics

Description

This command has been deprecated as this functionality is now included by default in Allocator.

Procedures Backward Compatibility

The previous section described the procedures that may be used in the Allocator configuration file. The new procedures are under the Tcl namespace LA; they should be prefixed with LA::

 **Note:** The Allocator configuration commands are different from the configuration commands used in the older Tcl version of the product. The older commands have been deprecated. For backwards compatibility, They will continue to function in this release.

The following table describes the mapping from the old command to the new command. It is recommended to migrate your `config.tcl` files to the new command set, as the old commands will be removed in a future version.

Old Command	New Command
addVovFlexImd	LA::AddLicenseMonitor
addLicenseMonitor	LA::AddLicenseMonitor
VovMQMapLSFResource	LA::MapLSFResource
getLSFResourceName	LA::GetLSFResourceName
addResource	LA::AddResource
VovMQAddResource	LA::AddResource
defineResourceGroup	LA::DefineResourceGroup
VovMQDefineResourceGroup	LA::DefineResourceGroup
addAllResources	LA::AddAllResources
VovMQAddAllResources	LA::AddAllResources
VovMQDisableQueue	LA::DisableSite
VovMQEnableQueue	LA::EnableSite
addQueue	LA::AddSite
VovMQAddQueue	LA::AddSite
defineUserMap	LA::DefineUserMap
MQmapUser	LA::MapUser
defineRemoteResourceName	LA::DefineRemoteResourceName

Old Command	New Command
VovMQNormalizeQueueName	LA::NormalizeSiteName
VovMQRegisterScript	LA::RegisterScript
VovMQGetMapForResourceInQueue	LA::GetMapForResourceInSite
VovMQSetMapForResourceInQueue	LA::SetMapForResourceInSite
setStandByQuantity	LA::SetMinQuantity
VovMQSetStandByQuantity	LA::SetMinQuantity
VovMQSetMinQuantity	LA::SetMinQuantity
VovMQSetMaxQuantity	LA::SetMaxQuantity
setReserveForUser	LA::SetReserveForUser
VovMQSetReserveForUser	LA::SetReserveForUser
VovMQSetResourceWeight	LA::SetResourceWeight
VovMQGetQueues	LA::GetSites
VovMQSetRank	LA::SetRank
VovMQSetSleepTimeInLoop	LA::SetSleepTimeInLoop
VovMQSetMinExpirationTime	LA::SetMinExpirationTime
VovMQGetQueueField	LA::GetSiteField
VovMQGetResourceField	LA::GetResourceField
VovMQGetResourceWeight	LA::GetResourceWeight
VovMQGetResourcePriority	LA::GetResourcePriority
VovMQGetResourceIsCritical	LA::GetResourceIsCritical

Allocator Usage Examples

This section contains simple examples about using Allocator to solve common problems of sharing licenses between sites.

In the following examples, the sites are called A B C D and the licenses to be shared are called x y z. It is assumed that all features are "WAN-able", that they can be shared across all sites.

A `config.tcl` file for Allocator typically contains elements from all the examples below.

Share One License Between Two Sites

In this example, the feature x coming from FlexNet Publisher is represented by the resource License:x. If there is contention between the two sites, more licenses will be allocated to B than to A because the weight for B is 200 and the weight for A is 100, so the ratio of the allocation will be 2 to 1 in favor of B. It is to be hoped that there are at least 3 tokens to share, otherwise the rounding errors may affect the sharing.

```
LA::AddResource License:x FLEXLM/x ""

LA::AddSite vnc@hostA A {} -version SAME
LA::AddSite vnc@hostB B {} -version SAME

LA::SetResourceWeight A License:x 100
LA::SetResourceWeight B License:x 200
```

Below is an equivalent way of setting the weights directly in the `AddSite` procedure. This is an older style.

```
LA::AddResource License:x FLEXLM/x ""

# Older way of specifying the weights.
LA::AddSite vnc@hostA A {
    License:x 100
}
LA::AddSite vnc@hostB B {
    License:x 200
}
```

Share One License Between Two Sites, with Minimum Allocations

In this example, even if there is no demand in site A, there will always be at least 2 tokens allocated to A. Similarly, 3 tokens are allocated to B regardless of the demand. We assume that there are at least 5 tokens to share.

```
LA::AddResource License:x FLEXLM/x ""

LA::AddSite vnc@hostA A {}
LA::AddSite vnc@hostB B {}

LA::SetMinQuantity A License:x 2
LA::SetMinQuantity B License:x 3
```

Share One License that Comes From Two Sources

In this example, the feature x is provided by two FlexNet Publisher daemons, identified by the tags TAG1 and TAG2 in Allocator. We do not care about the individual pools of licenses, but we want the aggregate number of licenses to be balanced between sites A and B.

```
LA::DefineResourceGroup License:x {  
  LA::AddResource License:TAG1_x FLEXLM/TAG1/x ""  
  LA::AddResource License:TAG2_x FLEXLM/TAG2/x ""  
}  
  
LA::AddSite vnc@hostA A {}  
LA::AddSite vnc@hostB B {}
```

The resource "License:x" is called the "group" while "License:TAG1_x" and "License:TAG2_x" are called the "components of the group". The resource "License:x" maps to the following expression:

```
"License:TAG1_x OR License:TAG2_x"
```



Note: When a feature comes from multiple daemons, remember to use the VOV_LM_VARNAME variable so that Accelerator can automatically set LM_LICENSE_FILE to point to the correct daemon.

Do Not Share with One Site

Sometimes one license cannot be shared with one of the sites, such as for contractual reasons. In this example, the license with tag TAG2 can be shared between sites A and B, but not with C.

```
LA::DefineResourceGroup License:x {  
  LA::AddResource License:TAG1_x FLEXLM/TAG1/x ""  
  LA::AddResource License:TAG2_x FLEXLM/TAG2/x ""  
}  
  
LA::AddSite vnc@hostA A {}  
LA::AddSite vnc@hostB B {}  
LA::AddSite vnc@hostB C {}  
  
LA::SetResourceWeight C License:TAG2_x DO_NOT_SHARE  
LA::SetResourceWeight C License:TAG1_x FROM_SITE
```

Reserve Some Licenses for Use Outside the Queue

Sometimes it is acceptable for some licenses to be used outside the queues. For example, user john may be working on a special project and he is running on a machine that is not managed by the site scheduler. We always want to make sure that john has at least 6 licenses reserved for this project.

```
LA::AddResource License:x FLEXLM/x ""  
  
LA::AddSite vnc@hostA A {}  
LA::AddSite vnc@hostB B {}  
  
LA::SetReserveForUser License:x john 6
```

Change Weights Depending on Job Priority

If a site has high priority jobs that are SCHEDULED and waiting for resources, then we may want to increase the weight for that site. The priority of a jobs in queue for a site can be retrieved with `LA::GetResourcePriority` as in the following example:

```
LA::AddResource License:x FLEXLM/x ""

LA::AddSite vnc@hostA A {} -access vtk
LA::AddSite vnc@hostB B {} -access vtk

proc myUpdateWeights {} {
  foreach site { A B } {
    set pri [LA::GetResourcePriority $site License:x]
    if { $pri >= 8 } {
      LA::SetResourceWeight $site License:x 500
    } else {
      LA::SetResourceWeight $site License:x 100
    }
  }
}

# The registered script will be evaluated once for each cycle.
LA::RegisterScript "myUpdateWeights"
```

Change Weights Depending on Time with TIMEVAR

We want to use different weights for sharing depending on the time of day. In this simple case, we just change weights on Saturday and Sunday.

```
LA::AddResource License:x FLEXLM/x ""

LA::AddSite vnc@hostA A {}
LA::AddSite vnc@hostB B {}

TIMEVAR changeWeightsOnWeekend {
  Sat,Sun {
    LA::SetResourceWeight A License:x 100
    LA::SetResourceWeight B License:x 100
  }
  default {
    LA::SetResourceWeight A License:x 400
    LA::SetResourceWeight B License:x 100
  }
}
```

Share All Licenses between Three Sites

```
LA::AddAllResources -resmapType License

LA::AddSite vnc@hostA A {} -version SAME -defaultweight 300
LA::AddSite vnc@hostB B {} -version SAME -defaultweight 200
LA::AddSite vnc@hostC C {} -version SAME -defaultweight 200
```



Note: Allocator provides the ability define resource groups which are a logical OR of components resources potentially hosted on different license servers. Further, licenses from specific servers can be excluded from distribution to specific sites.

Map Host Names

In several points in the systems, it is useful to get the host name for a checkout, for a job, for a daemon, etc. Since every machine may be known by different names, it is important to provide the expert user with the ability to map host names to some canonical form of the host name.

This is accomplished by overriding the procedure `mapHostName` which in its default implementation simply returns the short host name in all lower case. (Note, a fully qualified hostname is converted to its short name.)

```
# Default definition of mapHostName in vovutils.tcl
proc mapHostName { host { defaultName "" } { context "" } } {
    set shortHost [ string tolower [ lindex [split $host .] 0 ] ]
    return $shortHost
}
```

The procedure can be overridden using the file `hostmap.tcl` in the `.swd` directory. This file is read by calling the procedure `mapHostNameInit`.

```
mapHostNameInit <context>
```

where `context` is an optional parameter to specify any string that sets the context of the host mapping calls. If no `context` is provided, it defaults to an empty string.

Further, a `context` can also be specified as an argument to `mapHostName` procedure.

The default implementation of `mapHostName` does not use the `context`. However, a custom implementation specified in `hostmap.tcl` may either use the `context` provided in calling it, or it can access the context set in `mapHostNameInit` by using the variable `vovutils(hostmap,context)`:

```
# Example of hostmap.tcl
proc mapHostName { host { defaultName "" } { context "" } } {
    global vovutils
    switch -glob -- $host {
        "lnxBG*" { return [string tolower $host] }
        "lnx*.company.com" { return [lindex [split $host .] 0] }
        "localhost" {
            if { $vovutils(hostmap,context) eq "LM" } {
                return $defaultName
            } elseif { $context eq "NC-USA" } {
                return "mynchostname"
            }
        }
        default { return $host }
    }
}
```

The `mapHostName` procedure is used in Allocator:

- `vovlalm`: Used to sample Monitor data from Allocator. This script provides the nickname of the Monitor instance as the `context` on `mapHostNameInit`, and also uses the Monitor instance's nickname as the context in the call to `mapHostName`.

- `vovlavltnkncget` (called by `vovlanc`): Used to sample Accelerator data from Allocator. This script provides the nickname of the Accelerator site as the context on `mapHostNameInit`, and also uses the Accelerator instance's nickname as the context in the call to `mapHostName`.

Also to be noted, in Monitor, is the use of the built-in procedure `LMnormalizedName`, which is used to eliminate non-ASCII characters from user names, host names, and feature names.

Resource Groups

The notion of a "resource group" in Allocator is useful when you have a resource that is derived from multiple sources.

For example, you have some licenses of spice in India and some other license of the same spice in Russia, and you want to manage the aggregate of those license through Allocator.

Continuing the example, we are defining a resource group called "WAN:Spice" consisting of the two resources from the two servers, as follows:

```
#
# Fragment of vovlad/config.tcl
#

LA::DefineResourceGroup "WAN:Spice" {
    #
    # Two components in this group.
    #
    LA::AddResource "India:spice" FLEXLM/IN/spice ""
    LA::AddResource "Russia:spice" FLEXLM/RU/spice ""
}
```

This results in a resource map called "WAN:Spice" which is derived from the "OR'ing" of the licenses from India and the licenses from Russia, in addition to the individual resources called India:spice and Russia:spice.

By defining resource groups, Allocator can take into account the out-of-queue usage for the components when computing the allocation of the group.

It is possible to restrict distribution of a component to specific sites, such as geographically, by setting their weights to DO_NOT_SHARE.

```
#
# Fragment of vovlad/config.tcl
#

LA::SetResourceWeight "RU" "India:spice" DO_NOT_SHARE
```

Allocator and TIMEVAR

To make the weights of Allocator change as a function of time, you can use the procedure `TIMEVAR`.



Note: Allocator will set the weight of the components of a resource group from the weight of the resource group, unless the weight of the component has been set explicitly.

In the `vovlad/config.tcl` file you can add the `TIMEVAR` procedure (or procedures) at the bottom of the file. The `TIMEVAR` is then evaluated at every Allocator cycle (typically 30 seconds).

In the following example, the weight of the queues for a given resource is changed based on the day of the week.

```
#
# Fragment of vovlad/config.tcl
#
LA::AddResource Wan:drc 10 ""

...

TIMEVAR change_drc {
    Sat,Sun {
        LA::SetResourceWeight US    Wan:drc    30
        LA::SetResourceWeight IND  Wan:drc    100
    }
    default {
        LA::SetResourceWeight US    Wan:drc    50
        LA::SetResourceWeight IND  Wan:drc    50
    }
}
```

Passive Mode

Allocator can be run in "passive mode": it can do everything it needs to do except for controlling the resources of the sites. Using "passive mode" allows organizations to familiarize themselves with Allocator in their environment prior to deploying it.

To activate this passive mode, set the property LA_PASSIVEMODE to 1.

```
% vovprop set 1 LA_PASSIVEMODE 1
```

Two methods to disable "passive mode"

```
% vovprop del 1 LA_PASSIVEMODE  
% vovprop set 1 LA_PASSIVEMODE 0
```

License Overbooking

Advantages of Overbooking


If there are 10 licenses of a simulator, the scheduler (that is, Accelerator) dispatches 10 jobs using those licenses. However, many jobs do not use the licenses for 100% of their lifespan. If one checks how many licenses are checked out at any one time, for example with `lmstat`, one may find out that occasionally there are less than 10 licenses in use. Experienced users who look directly at the license daemon statistics may wonder why there are jobs in the queue while licenses are available. From Accelerator's point of view, those licenses are not available because they are reserved for those running jobs, which may check out the licenses at any time.

This problem is greatly amplified if instead of 10 licenses there are 1,000 licenses. In such a case, you may notice that 1,000 licenses are never fully checked out, although there are 1,000 running jobs at all times. For example, one customer had about 2,000 licenses of a simulator and even with 2,000 running jobs, only 1,750 to 1,900 licenses were checked out.

Those unused licenses create an opportunity to run more jobs than licenses, which is accomplished by "license overbooking".

Activate Overbooking

For overbooking to work well, you have to activate *vendor-queueing* for the license. If a job cannot find a license, it waits for the license to become available instead of failing. Allocator and Accelerator are capable of overbooking licenses so that only a small number of running jobs are waiting for a license.

 **Note:** The activation of vendor queueing is application dependent.

Overbooking Operation

The normal approach for managing licenses in Accelerator is setting the maximum number of licenses equal to the number available from the license daemons. Allocator and/or Accelerator issue jobs using this fixed number as an upper bound. In overbooking, tell Allocator and Accelerator to keep issuing jobs until it sees the actual license count (from the license daemon) fully depleted. The overbooking function acts as a classic control loop:

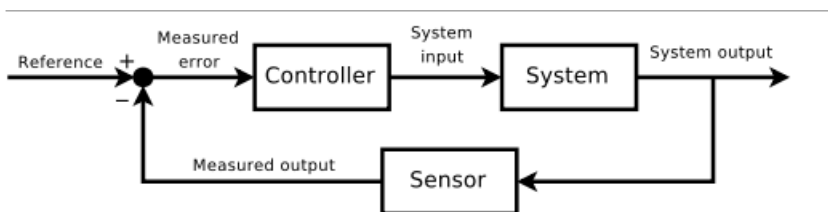


Figure 1:

The **Reference** is the number of licenses available, the **Measured output** is the number of licenses in use and the **Measured error** is the difference in the two. The **Controller** is the actual overbooking function that converts the license available into the number of jobs to issue - this is the **System input**. The **System output** is the launched jobs and the **Sensor** is the Monitor.

The Overbooking function (Controller) is `vtk_flexlm_overbook` for `vovresourced` and `LA::AddResource` in Allocator. These procedures have a number of tuning parameters for overbooking.

Overbooking in Allocator

The Overbooking function (Controller) is `vtk_flexlm_overbook` for `vovresourced` and `LA::AddResource` in Allocator. These procedures have a number of tuning parameters for overbooking:

<code>-thresh real</code>	The threshold at which overbooking becomes active. For example if the value is set to 0.8 it means that when the job count reaches 80% of the allowed total, overbooking starts increasing the number of submitted jobs. The default value for the parameter is 0.9.
<code>-factor real</code>	This number is used to scale the Measured Error into the additional number of jobs to submit. Common values are 0.8. The default value for this parameter is 1.0.
<code>-maxfactor real</code>	This number represents the maximum overbooking multiplicative growth factor on top of the existing licenses that is desired. For example if 10 licenses are available and we want to use overbooking to grow them to 100, the <code>maxfactor</code> should be set to 9.0 ($10 + 9.0 \times 10 = 100$) The default value for this parameter is 2.0.
<code>-headroom int</code>	When the license in use counts is greater than the maximum number of licenses less the headroom, the overbooking quantity is throttled. Negative headroom values are often used. The default value of headroom is 0.
<code>-queued int</code>	When Allocator sees this number of licenses queued (ie in vendor-queuing) then the overbooking quantity is throttled. Generally this should be a small number. The default value is 1.
<code>-lowpass int</code>	To smooth the effect of overbooking, the actual correction is low-pass filtered. This option controls the delay of the filter. The larger the number, the longer the delay and the smoother the correction. The default value is 8.
<code>-enable BOOL</code>	Simple way to enable/disable an overbooking rule
<code>-verbose BOOL</code>	Increase verbosity
<code>-v</code>	Same as <code>-verbose 1</code>

To activate overbooking, use either the browser interface for a resource or call:

```
vtk_mq_set_resource_field License:abcd overbook 1
```

Please note that in Allocator, activating overbooking implies disabling handle/job matching.

Similarly, to disable overbooking, you can use:

```
vtk_mq_set_resource_field License:abcd overbook 0  
vtk_mq_set_resource_field License:abcd match 1
```

To configure the overbooking parameters (see [License Overbooking](#) for details) you also need to use the same vtk interface.

Example:

```
vtk_mq_set_resource_field License:abcd overbook.thresh 0.9  
vtk_mq_set_resource_field License:abcd overbook.boostfactor 1.0  
vtk_mq_set_resource_field License:abcd overbook.maxfactor 2.0  
vtk_mq_set_resource_field License:abcd overbook.headroom 0  
vtk_mq_set_resource_field License:abcd overbook.vqthresh 1  
vtk_mq_set_resource_field License:abcd overbook.lpf.n 8; # Length of low-pass  
filter.
```

In Allocator, these settings have immediate effect. You can store them in your `vovlad/config.tcl` file.

Tuning Overbooking

The optimal values for the overbooking parameters are workload specific. While the default values work in many cases, it is worthwhile to review the overbooking operation. Use the `-verbose 1` option to monitor the behavior of the overbooking routines in the resource daemon log file. The goal should be to increase the license utilization to near 100% but without pushing too many jobs into vendor queuing. Vendor queuing implies that a hardware slot is taken but the job is stalled waiting for that license; normally a few slots being idle for a few seconds is a reasonable trade-off for high license utilization. However administrators should also be aware that having too many jobs in vendor queuing can cause the license daemons to stall, dramatically reducing overall throughput.

If your workload ends up pushing too many jobs into vendor queuing before backing off, even with a small value of `-queued` then you should consider reducing factor from 1.0 to say 0.9 (or lower) and increasing headroom from 0 to perhaps 2-5% of the maximum license quantity.

If your workload struggles to get any vendor queuing, it may be because the threshold is not reached or maintained - particularly when the new total (maximum number of jobs) has been increased. Reducing the threshold to 0.8 or 0.7 will enable overbooking to continue to be active. For a large number of short duration jobs, there can be many jobs "in-flight": jobs for which license matching by Accelerator or Monitor has not yet occurred. In these cases it may make sense to increase `-factor N` to say 1.2 and have a negative value of headroom equal to say 2-5% of the total license count.

While overbooking exhibits the self compensation characteristics expected from a control loop, tuning the parameters is often worthwhile for optimal usage.

Default values were used in the example above, which can be set to any other value using these commands.

Debugging

Since the behavior of Allocator is complex, facilities have been implemented to help you understand the behavior of the system.

You can start `vovlad` on the command line with multiple verbose options `-v`. Example:

```
% vovproject enable mq
% cd `vovserverdir -p vovlad`
% vovlad -v -v -v
```

When you start a new `vovlad` daemon, the old one automatically stops.

If you want to focus on a specific resources, use the Tcl array `MQdebug()` to specify the resource to debug. For example, if you want to study the behavior of Allocator with the resource "Wan:spice", you will write:

```
# Fragment of vovlad/config.tcl
set MQdebug(Wan:spice) 1
```

To stop debugging, use one of the following methods:

```
# Fragment of vovlad/config.tcl

## EITHER:
set MQdebug(Wan:spice) 0

## OR:
catch {unset MQdebug(Wan:spice)}
```

To enable and disable debugging on a specific resource, click the **Enable Debug** button on Allocator's web interface.

View Allocation Details

Web Interface

The primary access to Allocator is through its web browser interace.

Overview Page

After logging in to the browser interface, the first page presented is the **Overview** page. On this page, you can see a table of the resources being managed by this instance of Allocator.

Using the menus, you can choose to view either All Resources, or only Active Resources. Active resources either have jobs running or queued against them; jobs running could be through an Accelerator or out-of-queue.

For each resource, the table below shows a summary of the resource usage across all sites, as well as the usage on each individual site.

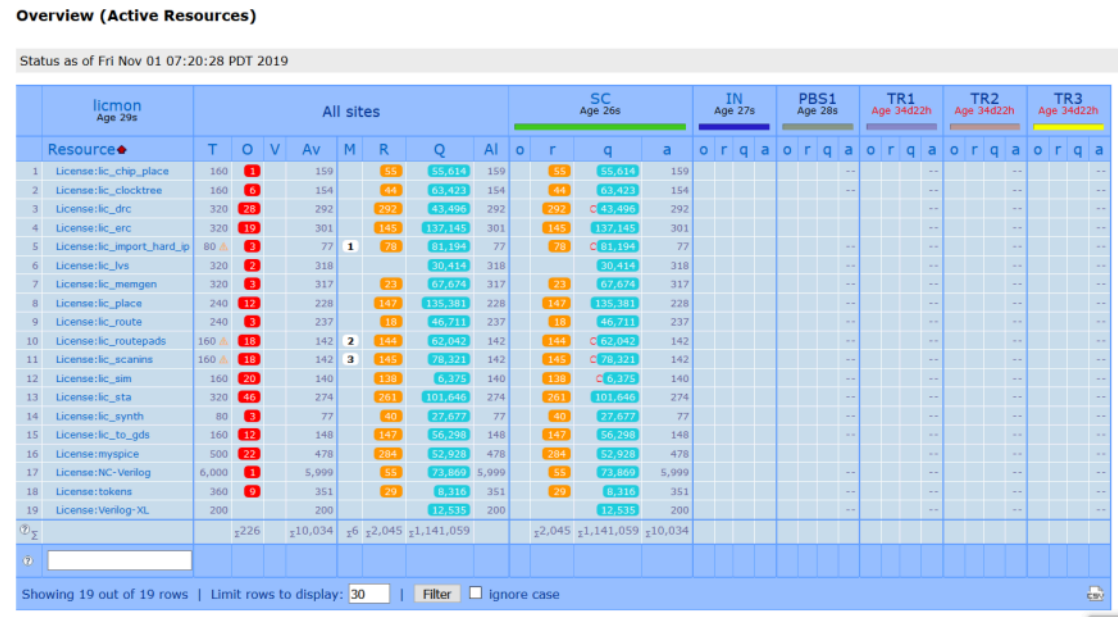


Figure 2:

The summary includes:

- (T) Total number of tokens for this resource
- (O) Number of tokens of this resource consumed by out-of-queue jobs
- (V) Number of vendor queued tokens for this resource
- (M) Number of tokens moving from one site to another that are currently in use, hence not available for distribution yet
- (A) Number of tokens available for distribution to various sites

Further, the table shows four columns for each site. The columns display:

- (o) Number of tokens of this resource consumed by out-of-queue jobs on this site
- (r) Number of tokens currently used by running jobs on this site
- (q) Number of tokens currently requested by queued jobs on this site
- (a) Number of tokens allocated by Allocator to this site

Links

The Overview page contains many links.

- Each resource name is a link to the detailed page for that resource. This page is described in the next section.
- In the first row first column, there are links to all Monitor instances this Allocator is interacting with. Clicking on any of these links will open the browser interface for that Monitor in a separate tab (or window).
- Each site heading is a link to that Accelerator's browser interface, which also opens in a separate tab (or window).
- The numbers in the running or queued jobs columns are links that take you to the appropriate page in Accelerator.

Resource Summary page

Clicking on a resource name on the **Overview** page takes you to that resource's details page. This page provides many details about the resource, and includes a table that shows how this specific resource has been allocated across different sites, and the various relevant attributes that were used to compute this allocation, such as its Weight, the Min and Max settings, Target allocations, etc. This page also includes a link to the Resource Map details for this resource.

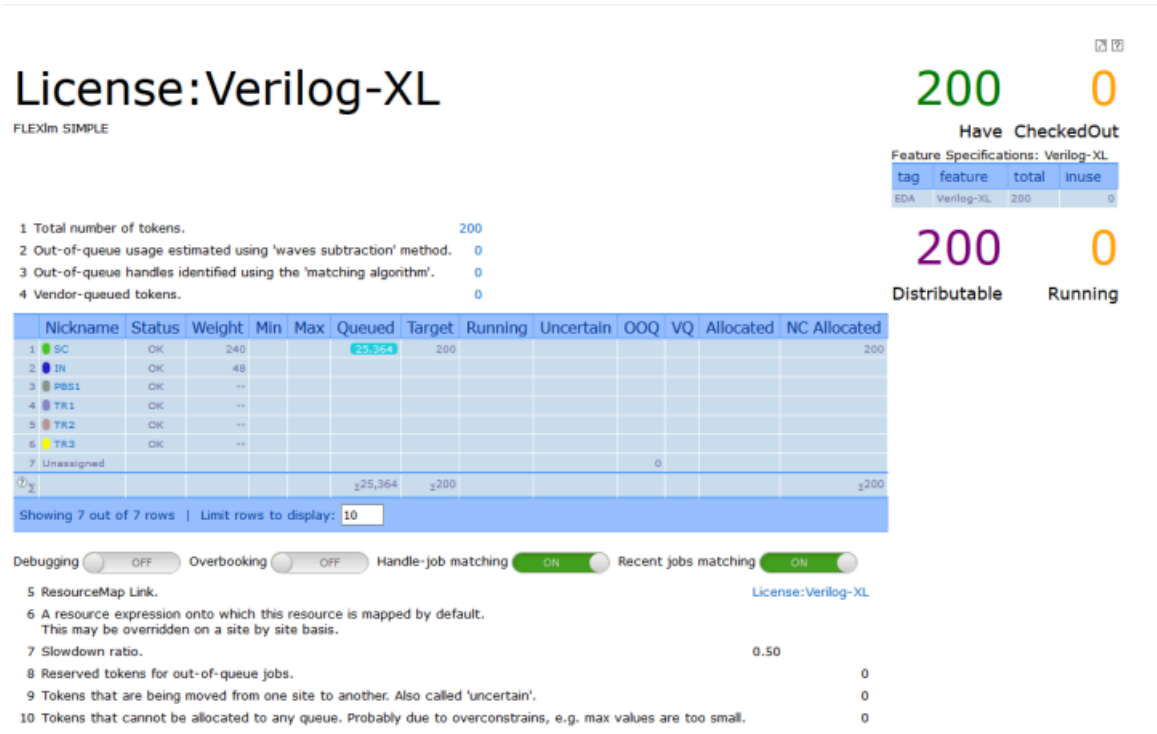


Figure 3:

If the resource displayed is a resource group, this page also lists out all the components of the group, and important numbers for each component. The name of each component is also a link which takes you to the details page of that component.

This page identifies if the resource is a simple resource, a resource group, or a component of a resource group.

Overbooking

Using this page, you can enable or disable overbooking on FlexNet Publisher type of resource. Also, you can choose to enable or disable handle to job matching for this resource. Note, matching can only be done if overbooking is disabled. Enabling overbooking will automatically disable matching.

Debugging

This page also provides the ability to enable or disable debugging for the resource. When debugging is enabled, the page will show a box containing intermediate numbers used for computing the allocation of this resource to various sites. These numbers are not meant for the user to interpret, but meant to be sent to Altair if the allocations or other numbers seem unexpected.

Resource Plots Page

This page shows a time plot of three metrics for this resource. The plots are:

- Running
- Allocated
- Queued

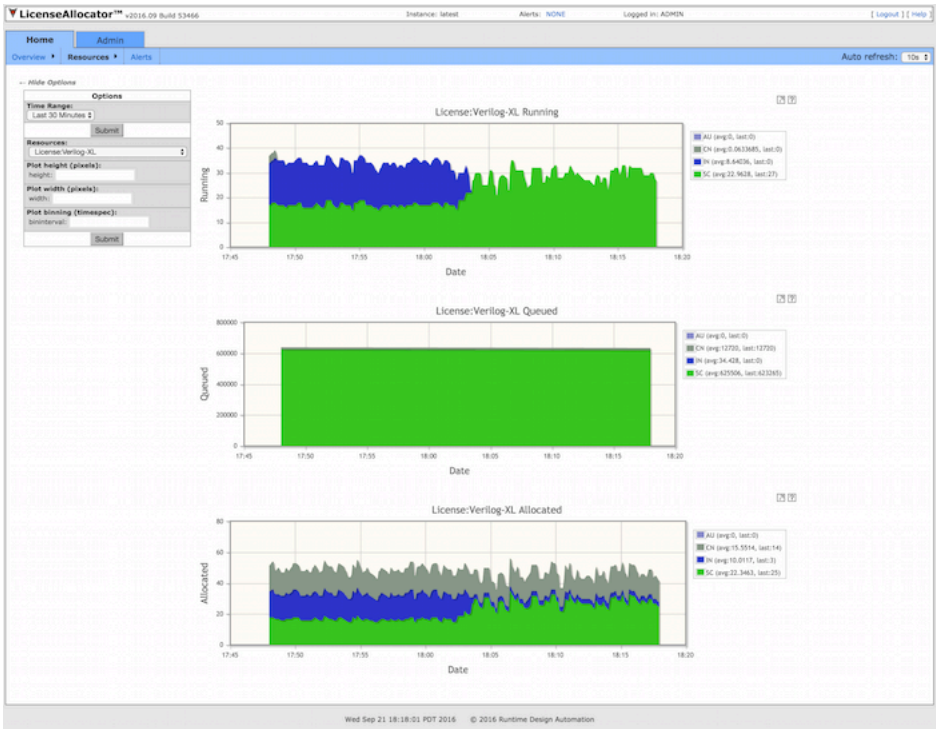


Figure 4:

The legend on the right of each plot tells what color is assigned to each site. Also, it is possible to hide the plot for a given site by clicking on the site in the legend.

Admin Page

The Admin page can be used to monitor the periodic jobs that are used to sample Monitor and Accelerator instances that Allocator has been connected to.

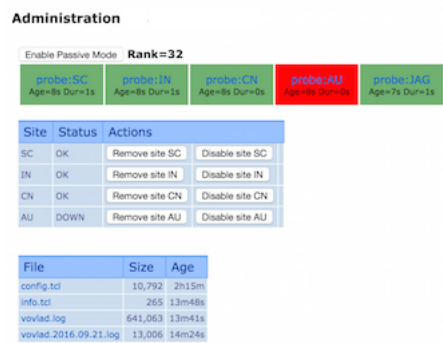


Figure 5:

Site Management

You can choose to enable or disable a specific site from participating in the License Allocation, or even choose to completely remove the site. Note, if you remove a site using the browser interface, you cannot add it back using the browser interface. You will need to go back to the `config.tcl` file and touch it to force Allocator to reread it and restore the site.

Passive Mode

Allocator can be run in **passive** mode. In this mode, Allocator will continue to sample Monitor and Accelerator and performs all allocation computations. However, it will no longer send the allocations back to the Accelerator instances. This is a mechanism for you to run Allocator in a **read only** fashion to monitor its computations without affecting a running production Accelerator instance.

Standalone User Interface

Allocator also provides a standalone graphical user interface that can be run from the shell command line.

Please note, this is a limited GUI which can show details of only one resource at a time. For full UI functionality, it is recommended to use the browser interface, described in an earlier section.

In a shell, use the command:

```
vovla_gui -r <RESOURCE_NAME>
```

This standalone GUI will display four time plots for the specified resource for each site:

- Running
- Queued
- Allocated
- Uncertain



Figure 6:



Note: Each site uses a different color to distinguish it from other sites. However, the chosen color for any site is the same across all four plots. For any of the plots, you can hide the plots for one or more sites by clicking on the site nickname(s) in the left column.

On the right side of the screen, you can see the average value for each plot as well as the last value.

The plots shown here auto-refresh every 10 seconds by default. You can choose the "update" interval by clicking the **update** button in the top right and selecting the desired refresh rate from the drop-down menu.

You can also choose the time range to display in these plots by clicking the **interval** button and choosing the desired time range from the drop menu.

Command Shell Output

Allocations made by Allocator and other details related to the allocation can be viewed on the command line of a shell where the Allocator project has been enabled.



Note: This is a very raw output. To get a more useful view and functionality, it is recommended to use the browser interface, described in an earlier section.

In a shell, use the following command:

```
vovshow -la
```

This will display the allocation details in a tabular format, as shown below. For each resource, this output will give a total summary across all sites, as well as details of individual sites identified by their nickname. For each site, it shows 3 values as r/a/q: running/allocated/queued.

```
% vovshow -la
LMSITE 000097801 licmon
NCSITE 000103532 SC
NCSITE 000106903 IN
NCSITE 000110274 CN
RESOURCE_NAME
CN r/a/q
License:111
7 0 0 0/ 6/ 0 0/ 1/ 0
0/ 0/ 0
License:940
1 0 0 0/ 1/ 0 0/ 0/ 0
0/ 0/ 0
License:945
1 0 0 0/ 1/ 0 0/ 0/ 0
0/ 0/ 0
License:AAA
62 0 0 0/ 52/ 0 0/ 10/ 0
0/ 0/ 0
License:AEMv8_RAVEN
400 0 0 0/334/ 0 0/ 66/ 0
0/ 0/ 0
License:AdvancedAMSDebug
3 0 0 0/ 3/ 0 0/ 0/ 0
0/ 0/ 0
License:Affirma_AMS_distrib_p
60 0 0 0/ 50/ 0 0/ 10/ 0
0/ 0/ 0
License:Affirma_sim_analysis_env
5 0 0 0/ 5/ 0 0/ 0/ 0
0/ 0/ 0
```



```
...  
...
```

You can choose to see only a specific column for each site by specifying the name of that column on the `vovshow` command line. The available columns are `inuse`, `alloc`, `queued`, or `all`. If no column is specified, it defaults to showing all columns.

```
vovshow -la alloc
```


You can see details of only a specific resource as shown below:

```
vovshow -lares License:workflowsim
```

Failover Server Candidates

If a server crashes suddenly, VOV has the capability to start a replacement server on a pre-selected host. This capability requires that the pre-selected host is configured as a failover server.

The configuration instructions follow.

 **Note:** The `vovserverdir` command only works from a VOV-enabled shell when the project server is running.

1. Edit or create the file `servercandidates.tcl` in the server configuration directory. Use the `vovserverdir` command with the `-p` option to find the pathname to this file.

```
% vovserverdir -p servercandidates.tcl  
/home/john/vov/myProject.swd/servercandidates.tcl
```

The `servercandidates.tcl` file should set the Tcl variable `ServerCandidates` to a list of possible failover hosts. This list may include the original host on which the server was started.

```
set ServerCandidates {  
    host1  
    host2  
    host3  
}
```

2. Install the `autostart/failover.sh` script as follows:

```
% cd `vovserverdir -p .`  
% mkdir autostart  
% cp $VOVDIR/etc/autostart/failover.sh autostart/failover.sh  
% chmod a+x autostart/failover.sh
```

3. Activate the failover facility by running `vovautostart`.

```
% vovautostart
```

For example:

```
% vovtaskermgr show -taskergroups  
ID          taskername      hostname      taskergroup
```

000404374	localhost-2	titanus	g1
000404375	localhost-1	titanus	g1
000404376	localhost-5	titanus	g1
000404377	localhost-3	titanus	g1
000404378	localhost-4	titanus	g1
000404391	failover	titanus	failover



Note: Each machine listed as a server candidate must be a vovtasker machine; the vovtasker running on that machine acts as its agent in selecting a new server host. Taskers can be configured as dedicated failover candidates that are not allowed to run jobs by using the `-failover` option in the taskers definition.

Preventing jobs from running on the candidate machine eliminates the risks of machine stability being affected by demanding jobs. The `-failover` option also enables some failover configuration validation checks. Failover taskers are started before the regular queue taskers, which helps ensure a failover tasker is available as soon as possible for future failover events.

The `-failover` option keeps taskers up and running without the need for jobs to be running (in fact, jobs are not allowed to run on them). This allows them to participate in the server election process and start up vovserver without introducing a competition for resources.

Without `-failover`, the taskers will be normal ones, which can run jobs, and in fact, they *must* be running jobs at the time of the vovserver kill/crash because without no jobs, the taskers will exit:

```
if ( ss_activeTransitions <= 0  && (! isInFailoverGroup ) ) {  
    addLog( "No jobs running: no need to keep running." );  
    goto tasker_exit;  
}
```

The easiest way to ensure the tasker is in the failover group is to use the `-failover` option.

Refer to the tasker definition documentation for details on the `-failover` option.

How vovserver Failover Works

If the vovserver crashes, after a period of time, the vovtasker process on each machine notices that it has had no contact from the server, and it initiates a server machine election.


In this election, each vovtasker votes for itself (precisely, the host that this particular tasker runs on) as a server candidate. The election is conducted by running the script `vovservsel.tcl`.

After the time interval during which the vovtaskers vote expires, (default 60 seconds) the host that appears earliest on the list will be selected to start a new vovserver.

In the following example, the `servercandidates.tcl`, file contains three hosts:

```
set ServerCandidates {  
    host1  
    host2  
    host3  
}
```

When the server crashes, if there are vovtaskers running on host1, host2 and host3, then these hosts will be voted as server candidates. Then host2 will be the best candidate and a new vovserver will be started on host2. This server will start in crash recovery mode.

 **Note:** For failover recovery to be successful, an active `vovtasker` process must be running on at least one of the hosts named in the `ServerCandidates` list. Usually, these vovtaskers have been defined with the `-failover` option so they can not accept any jobs, and are members of the `failover` taskergroup.

The failover vovserver will read the most-recently-saved PR file from the `.swd/trace.db` directory, and then read in the transactions from the 'cr*' (crash recovery) files to recover as much of the pre-crash state as possible.

The vovserver writes a new `serverinfo.tcl` file in the `.swd` that vovtaskers read to determine the port and host. When it starts, the failover vovserver appends the new host and port information to the `$NC_CONFIG_DIR/<queue-name.tcl>` as well as to the `setup.tcl` in the server configuration directory. The vovserver then runs the scripts in the autostart directory. This should include the `failover.sh` script, which resets the failover directory so that failover can repeat. This script removes the registry entry, and removes the `server_election` directory and creates a new empty one. At the end, it calls `vovproject reread` to force the failover vovserver to create an updated registry entry.

The failover vovserver remains in crash recovery mode for an interval, usually one minute, waiting for any vovtaskers that have running jobs to reconnect:

- For Accelerator, Accelerator Plus, Monitor and Allocator, vovtaskers wait up to 4 days for a new server to start.
- For FlowTracer, vovtaskers wait up to 3 minutes for a new server to start.

After reconnecting to vovserver, vovtaskers automatically exit after all of their running jobs are completed. After the vovserver transitions from crash recovery mode to normal mode, it will try to restart any configured vovtaskers that are not yet running.

Any of the following conditions will prevent successful failover server restart:

- The filesystem holding the `.swd` directory is unavailable.
- The file `servercandidates.tcl` does not exist.
- The `ServerCandidates` list is empty.
- There is no vovtasker running on any host in the `ServerCandidates` list when the server crashes.
- The `autostart/failover.sh` script file is not in place.

In this case, the failover server will not be automatically started; the server will have to be manually started.

Tips for Configuring Failover

Following are tips for failover configuration:

- Make the first failover host the regular one. This way, if the vovserver dumps core or is killed by mistake, it will restart on the regular host.
- Configure special vovtaskers only for failover by passing the `-failover` option to `vtk_tasker_define`.

- Test that failover works before depending on it.

Migrating vovserver to a New Host

The failover mechanism provides the underpinnings of a convenient user CLI command that can be used to migrate vovserver to a new host:

```
ncmgr rehost -host NEWHOST
```

The specified NEWHOST must be one of the hosts eligible for failover of vovserver.

Multiple Queues

Queue Name and Host Considerations

When setting up multiple Accelerator queues, there are factors to consider:

- Of course, each host that runs a vovserver to manage an Accelerator queue needs access to the Altair Accelerator software. In most cases, this is done by automounting the software from a file server, but a host- local install may also be used.
- The name of the queue must begin with the letters 'vnc'. This indicates that it is an Accelerator vovserver, so that it will check out the correct license feature 'server_nc'.

It is helpful if the remainder of the queue name encodes something that identifies the queue. For example, 'vnscj' could represent a server running in San Jose, CA.

It is also required that the queue names are unique within those running on the same or a replicated Altair Accelerator software hierarchy. Two queues may not have the same name that run on the same Altair Accelerator software installation, even if you try to start them on different hosts.

An Accelerator queue is a specialized case of a VOV 'project', and the `ncmgr` command calls the `vovproject` command to start the vovserver that manages the queue. The latter command uses the Altair Accelerator registry to store data about all the known projects.

- The host and TCP/IP port combination must be unique for all queues. The default is to compute the port number by hashing the queue name into the range 6200-6455. You may specify the port number using the `-port` option when *first starting* the queue.

Create a New Queue

To create a queue with a given name, use the `-queue` option of the `ncmgr start` command.

For example, if you have already started the default queue 'vnc' and have it in production, and want to start a separate queue to test a new Altair Accelerator version, you might create a new queue called 'vnctest'. Example:

```
% ncmgr start -q vnctest
```

To configure the new vnctest queue, you could copy the configuration files from `vnc.swd` **except** `setup.tcl`. This should not be copied, because it will usually have a different port number, and *must* have a different port if running on the same host.

You would also want to edit the `taskers.tcl` file for the vnc826 queue, so that only a few hosts are in it for testing.

How the Default Queue is Determined

When you have multiple queues, the Accelerator commands act on the default queue when no other is specified. The Accelerator administrator can control the default using files in the `NC_CONFIG_DIR` directory (usually `$VOVDIR/local/vncConfig`).

The files in that directory are in Tcl format, and set environment variables used to determine the vovserver to which your Accelerator command sends RPC. Whatever is set by the one named `vnc.tcl` determines the default. The file for each queue has the form `<queue-name>.tcl`, and is created by

the `ncmgr` command when the queue is first started. A useful trick is to symbolic link from the queue-specific file to `vnc.tcl`, permitting the Accelerator admin to quickly and easily change the default.

Working with Multiple Queues

There are two ways to specify a non-default queue in Accelerator:

- The `-queue` command line option
- The `NC_QUEUE` environment variable

The Accelerator commands accept a `-queue` option to specify which queue to act on. This permits the queue to be selected on a command-by-command basis, but adds extra typing. You can abbreviate this to `-q`

If you will be working primarily with a queue other than the default 'vnc', it is better to set the environment variable `NC_QUEUE` to the name of the queue.

For example, suppose you have two sites in San Jose, and Andover, MA, and the queues are named 'vnscsj' and 'vncma' respectively. The Accelerator admin would set the default queue in San Jose to be 'vnscsj', and in Andover, to be 'vncma'.

A user in San Jose who wants to see the jobs of user 'carl' in Andover could do as shown below:

```
% nc -q vncma list -u carl
```

Instead of relying on the default, you can name the local queue explicitly:

```
% nc -q vnscsj list -u carl
```

Tradeoffs Separating Farm Hosts

There are trade-offs to separating farm hosts into separate queues. Some of them are:

- When you divide your compute farm hosts into separate queues, you limit the number of job slots users have to run jobs without specifying a non-default queue.
- More important, once a job is submitted to a queue, it stays there. So, jobs could wait longer if they are submitted to a loaded queue, but there are open slots on a different queue.
- Separate queues permit maintenance shutdowns without completely stopping batch queue service. Since the addition of the `-freeze` option to `ncmgr stop`, you can even replace the `vovserver` binary without needing to stop running jobs, making this less of a concern.

Queues Example

This section illustrates the interaction of Allocator with FlexNet Publisher. The setup consists of two Accelerator installations, one in Santa Clara, California, and one in Bangalore, India. We use the Allocator product to allocate resources among the two sites.

In this example we monitor the resource `License:lic_sim`. Upon querying Monitor, Allocator learns that we currently have a license to run up to 7 concurrent jobs using this resource. Under idle

conditions, this license is distributed to the two Accelerator sites as per the specified default weights for the sites.

```
# Fragment of vovlad/config.tcl
LA::AddSite vncsc@santaclara "SC" {
} -port 6229 -version 2016.09 -defaultweight 50

LA::AddSite vncblr@bangalore "BL" {
} -port 6335 -version 2016.09 -defaultweight 20
```

Using these default weights, Allocator will determine that the Santa Clara office gets access to 5 licenses, while the remote Bangalore office gets 2.

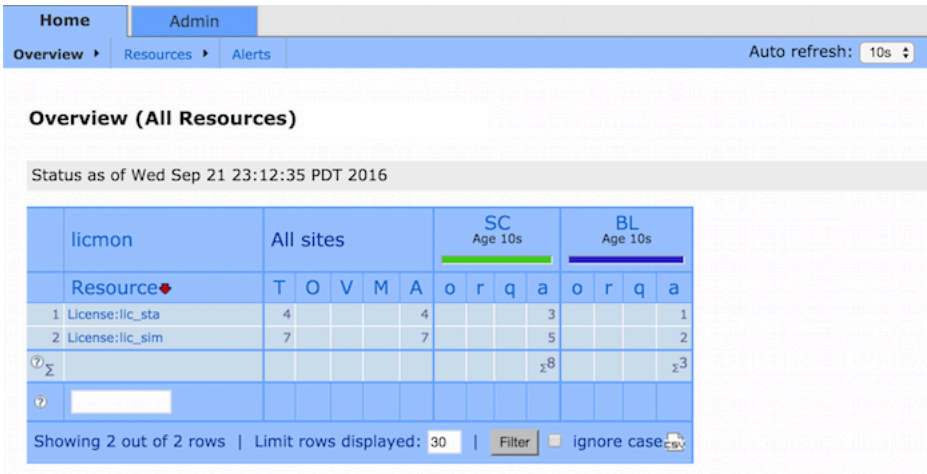


Figure 7:

When a user submits a large batch of jobs in Santa Clara, the licenses get quickly migrated from Bangalore to Santa Clara.

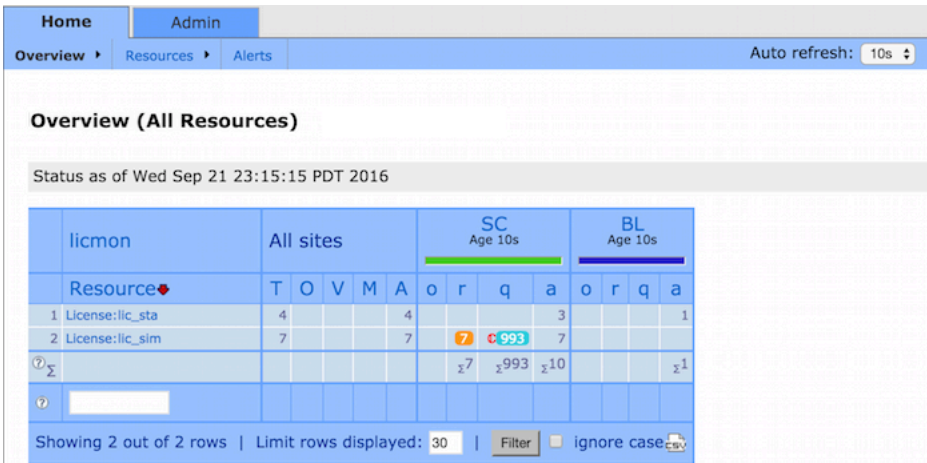


Figure 8:

Now Bangalore also requests to use the same resource. The system therefore wants to send 2 licenses back to Bangalore. This is done incrementally: as the jobs in Santa Clara complete, the license is moved to Bangalore.

The first picture shows allocation to Santa Clara reduced to 5, but since 7 jobs are still running, the 2 licenses cannot yet be moved to Bangalore until running jobs release them, so those 2 are shown in the "Moving" column.

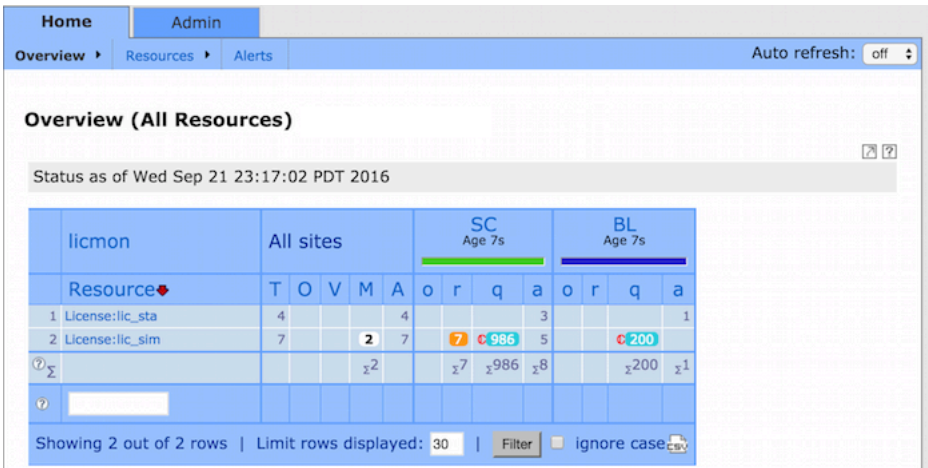


Figure 9:

Once the running jobs in Santa Clara drop to 5, the 2 freed up licenses are now moved to Bangalore and jobs start running in Bangalore.

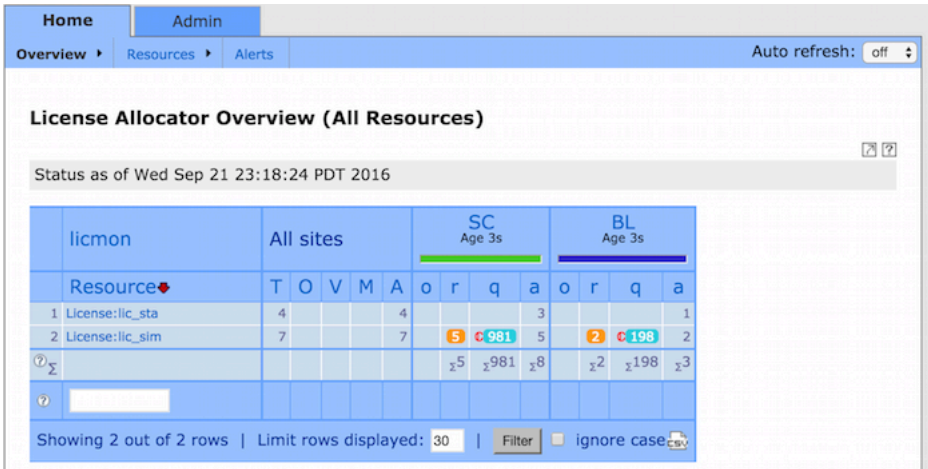


Figure 10:

The Bangalore workload is now zero, but a user in Bangalore (john@sile) is using one of the licenses outside of Accelerator, so the system needs to throttle down the number of licenses available for Santa Clara from 7 to 6.

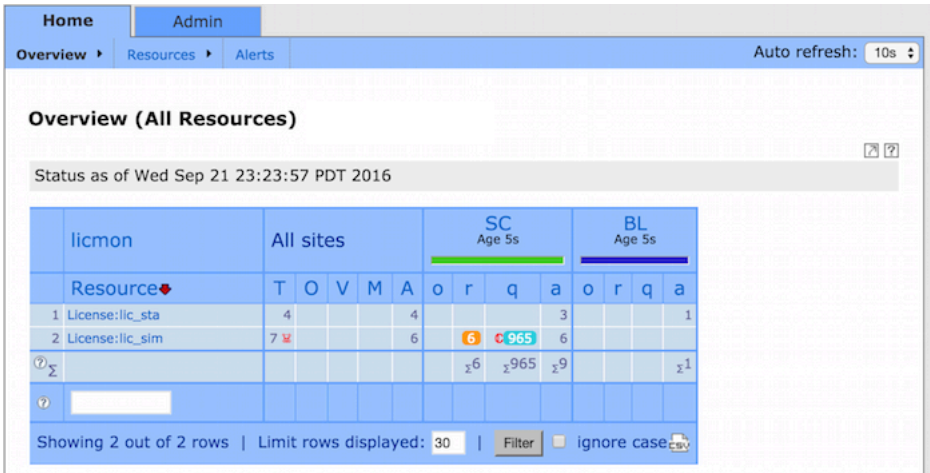


Figure 11:

By navigating the browser interface, we can view the historical plots for the utilization, unmet demand (queued), and allocation for this resource, say for the past 4 hours.

PBS and Other Queues

Allocator can allocate licenses across multiple schedulers, including Accelerator queues, PBS clusters and others. In this section we discuss how to integrate Allocator and PBS.

Preparing a PBS Environment

We need an environment called PBS to be available, at least, on the host where Allocator is running. An example of this environment could be this:

```
#  
# This should be $VOVDIR/local/environments/PBS.start.csh  
#  
set path = `vovenv PREPEND /opt/pbs/bin $path`  
setenv MANPATH `vovenv PREPEND -: /opt/pbs/share/man $MANPATH`  
setenv LD_LIBRARY_PATH `vovenv PREPEND -: /opt/pbs/lib $LD_LIBRARY_PATH`
```

This environment is needed by the command `vovlapbs` which probes the status of PBS and sends it to Allocator.

Enabling qmgr Access from the Allocator Server Machine

Allocator actively manages one or more PBS static resources that correspond to software licenses or license options. These resources are termed "LA-managed resources". Allocator manages and updates the resource availability in PBS by calling `qmgr` from the Allocator server machine. It is therefore important that the user running Allocator has manager privileges on the PBS instance.

In the following example, PBS is running on the machine `pbshost1`. Allocator is running as user `cadmgr` on the machine called `lahost1`:

```
pbshost1 # qmgr  
Max open servers: 49  
Qmgr: set server flatuid=true  
Qmgr: set server managers += cadmgr@lahost1.company.com ... Or...  
Qmgr: set server managers += cadmgr@*  
Qmgr: list server managers  
managers = cadmgr@*
```

Configuring the PBS Scheduler

All resources managed by Allocator must be declared like any other static resource in the "resources:" line in the `sched_config` file on the host that is running the PBS scheduler. In specific, you have to change the "resources:" line in `sched_config`. This file is typically in `/var/spool/pbs/sched_priv/sched_config`. In the following example, we add the three Allocator-managed resources `License_spice`, `License_lic_drc` and `License_lic_erc`. Notice that we cannot use the colon ":" in the license names, as is common in Accelerator.

```
# Example: fragment of sched_config  
resources: "ncpus, mem, arch, host, vnode, aoe, eoe, License_spice, License_lic_drc,  
License_lic_erc"
```

Then you have to tell the PBS scheduler to re-read the config file. You can do that by sending the signal HUP to the scheduler process, as in this example:

```
# cd /var/spool/pbs/sched_priv
# kill -HUP `cat sched.lock`
```

Finally, you need to use `qmgr` to create the resources and to declare them as schedulable by the queue:

```
# qmgr
qmgr: create resource License_spice type=long,flag=q
qmgr: create resource License_lic_drc type=long,flag=q
qmgr: create resource License_lic_erc type=long,flag=q

% qmgr -c "list resource"
```

Configuring Allocator

On the Allocator side, you have to add a site to the config file in `la.swd/vovlad/config.tcl`.

```
## Fragment of vovlad/config.tcl file
## We initially set the default weight to -1 (DO_NOT_SHARE)
## Then we selectively set the weight of various resources to the desired value.
LA::AddSite laval_NOTUSED PBS1 {} -scheduler PBS -remotehost laval -timezone PST8PDT
    -defaultweight -1

## Assign a weight to the resources that are actually shared with PBS
foreach lic { lic_drc lic_erc spice } {
    LA::DefineRemoteResourceName PBS1 License:$lic License_$lic

    LA::SetResourceWeight PBS1 License:$lic 75 ; # Use the "official" resource name,
    not the PBS name.
}
```

The underlying PBS queue needs to have job history tracking enabled. This is accomplished by the following PBS admin command:

```
% qmgr -c 'set server job_history_enable=true'
```


Using PBS with Allocator

There is no change in the way jobs are submitted to PBS. If the submitted job requires a license static resource that is managed by Allocator, then the availability of that license will change over time every few seconds depending on the workload at the various sites.

```
% qsub -l License_lic_drc=1 -- /bin/sleep 1000
% qsub -N DRC -J 1-10 -l License_lic_drc=1 -- /bin/sleep 1000
% qsub -l select=4:ncpus=1 -lplace=scatter -l License_vera=1 -- /bin/sleep
100
% qsub -J 1-10 -l select=4:ncpus=1 -lplace=scatter -l License_vera=1 -- /bin/sleep
100
```

Configure an LSF or OpenLava Queue

The configuration of an LSF or OpenLava queue uses the procedure LA::AddSite.

 **Note:** The LSF interface is no longer supported. The underlying API will be removed in a future release.

The following is a complete example of a configuration. Although the example uses LSF, it also applies to OpenLava.

The scenario is the following:

- We have an LSF master running on the machine uni00. We are going to use that same machine to run bjobs. We are going to login to that machine as user "cadmgr".
- The LSF master is running in the timezone called PST8PDT (US-LosAngeles)
- We are going to call the cluster "lsfUni"
- We are going to setup a directory to run bjobs and the location of such directory as seen from uni00 is /<remote_host_directory>/lsf_la_dir.

```
set lsfQueue          "lsfUni"
set laDirOnRemoteHost /some/dir/on/remote/host/lsf_la_dir
set remoteHost        "uni00"
set remoteUser        "cadmgr"
set lsfWeight         120
set lsfDur            0
set lsfTZ             "PST8PDT"

LA::AddSite lsf1@uni00 $lsfQueue {} \
  -lsf \
  -lsfdur      $lsfDur \
  -remotehost  $remoteHost \
  -user        $remoteUser \
  -lsfdir      $laDirOnRemoteHost \
  -defaultweight $lsfWeight \
  -timezone    $lsfTZ
```

Options

-lsf	LSF or OpenLava cluster, rather than an Accelerator instance.
-lsfdur	Controls the interpretation of "duration=N" inside of <code>rusage[]</code> statement. If the value of <code>-lsfdur</code> is 1, then the duration is honored, else it is ignored. The default is 0, i.e. to ignore the duration statement.
-remotehost and -user	Together specify a user,host pair used to run bjobs. The system needs passwordless ssh access to that host for that user.
-lsfdir	Specifies a directory that must exist on the remote host.
-defaultWeight	specifies the weight of this cluster relative to the weights of other LSF or OpenLava clusters or Accelerator instances. For more

information on weight, please see the documentation for the [LA::AddSite](#) command.

-timezone

Specifies the time zones to be used when running bjobs. This is normally the timezone of the Allocator server.

Steps Used to Sample LSF

Based on the above configuration, vovlad creates a periodic job that is used to to sample the LSF or OpenLava cluster and to import the data into Accelerator. This job looks as following:

```
vovlalsf $lsfQueue $removeHost $laDirOnRemoteHost $remoteUser $lsfDur $lsfTZ  
vovlabjobs.sh
```

This periodic jobs runs a few scripts to get the bjobs details from LSF. To set the allocations on LSF through elim:

```
vovlagetbjobs -host $remoteHost -out ../$lsfQueue.bjobs.tz -queue $lsfQueue \  
-u $remoteUser -dir $laDirOnRemoteHost -script vovlabjobs.sh
```

This is followed by another job that translates the output of bjobs into information suitable for Accelerator.

```
env TZ=$lsfTZ vovlatranslatebjobs -queue $lsfQueue -dur $lsfDur -f \  
../bjobs.last.gz -host $remoteHost -no-tz-correct
```

Preparing the Remote Directory where the Sampling is Done

On the remote machine that has access to "bjobs", you need to setup a directory with a few scripts to perform the periodic sampling of LSF. The templates for those scripts can be found in \$VOVDIR/etc/la/.

Copy the file \$VOVDIR/etc/la/vovlabjobs.sh to the remote directory and make it executable. It is perfectly normal to assume that you will have to edit the file to make it work in your environment.

The output of bjobs -l needs to be preprocessed by the script \$VOVDIR/etc/la/vovlapreprocessbjobs which should also be copied to the remote directory and should be made executable.

Configure an LSF or OpenLava elim

In order to configure LSF or OpenLava, you need to be an LSF expert. You need to know where your lsf.conf and lsf.shared files reside and how to configure "elims".



Note: The LSF interface is no longer supported. The underlying API will be removed in a future release.

Here is an example of one elim implemented as an infinite loop where a file generated by Allocator is read every few seconds:

```
#!/bin/csh -f  
  
while (1)
```

```
# The lsfUni.elim file is created by Allocator
/bin/cat /remote/cadmgr/lsf_la_dir/in.elim/lsfUni.elim
sleep 20
end
```

1. Specify the license resources in the file `lsf.shared` in the Resource section:

```
AstroLic           Numeric 10 N (FlexNet Publisher License Resource)
CalibredrcLic      Numeric 10 N (FlexNet Publisher License Resource)
```

2. Specify at least one elim shared license in ResourceMap section in the `lsf.cluster.<lsfClusterName>` file.

```
Begin ResourceMap
RESOURCENAME  LOCATION
tmp2          [default]
# nio         [all]
# console     [default]
#cpuf         [10@localhost]
#ncpus        [10@localhost]
AstroLic      [default]
End ResourceMap
```

3. To verify that your setup is working, check the following commands:

```
bhosts -l          #--- Displays available licenses if elim is working
                    correctly.
lshosts -R resource #--- List the hosts where the resources are available.
lsinfo            #--- Lists all resources.
```

For example:

```
% lshosts -R AstroLic
```

The "elim process" run all that time if everything is configured correctly.

Troubleshooting elims

Problem

Cannot start `lim`, in the `lim.log.<host_name>`, getting the following error message

```
setMyClusterName(): unable to find the cluster file
                    containing local host localhost
```

Solution

Make sure the host name in `lsf.cluster.<cluster_name>`, in host section, is the name that you are getting with `hostname -f` command. You can modify the `/etc/hosts` file to change this hostname.

Problem

`bhosts -s resource` (ex: `bhosts -s VerdiX`) is saying "Bad resource name"

Solution

Add the resource in the `lsf.cluster.java` file under ResourceMap section. Example:

```
Begin ResourceMap
RESOURCENAME    LOCATION
VerdiX          [all]
primetimeX      [all]
End ResourceMap
```

Legal Notices

Intellectual Property Rights Notice

Copyrights, trademarks, trade secrets, patents and third party software licenses.

Copyright ©1986-2025 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of the intellectual property rights held by Altair Engineering Inc. or its affiliates. Software, other products, and materials of Altair Engineering Inc. or its affiliates are protected under laws of the United States and laws of other jurisdictions.

In addition to intellectual property rights indicated herein, such software, other products, and materials of Altair Engineering Inc. or its affiliates may be further protected by patents, additional copyrights, additional trademarks, trade secrets, and additional other intellectual property rights. For avoidance of doubt, copyright notice does not imply publication. Copyrights in the below are held by Altair Engineering Inc. or its affiliates. Additionally, all non-Altair marks are the property of their respective owners. If you have any questions regarding trademarks or registrations, please contact marketing and legal.

This Intellectual Property Rights Notice does not give you any right to any product, such as software, or underlying intellectual property rights of Altair Engineering Inc. or its affiliates. Usage, for example, of software of Altair Engineering Inc. or its affiliates is governed by and dependent on a valid license agreement.

Altair HyperWorks®, a Design & Simulation Platform

Altair® AcuSolve® ©1997-2025

Altair® Activate® ©1989-2025

Altair® Automated Reporting Director™ ©2008-2022

Altair® Battery Damage Identifier™ ©2019-2025

Altair® CFD™ ©1990-2025

Altair Compose® ©2007-2025

Altair® ConnectMe™ ©2014-2025

Altair® DesignAI™ ©2022-2025

Altair® DSim® ©2024-2025

Altair® DSim® Cloud ©2024-2025

Altair® DSim® Cloud CLI ©2024-2025

Altair® DSim® Studio ©2024-2025

Altair® EDEM™ ©2005-2025

Altair® EEvision™ ©2018-2025

Altair® ElectroFlo™ ©1992-2025
Altair Embed® ©1989-2025
Altair Embed® SE ©1989-2025
Altair Embed®/Digital Power Designer ©2012-2025
Altair Embed®/eDrives ©2012-2025
Altair Embed® Viewer ©1996-2025
Altair® e-Motor Director™ ©2019-2025
Altair® ESAComp® ©1992-2025
Altair® expertAI™ ©2020-2025
Altair® Feko® ©1999-2025
Altair® FlightStream® ©2017-2025
Altair® Flow Simulator™ ©2016-2025
Altair® Flux® ©1983-2025
Altair® FluxMotor® ©2017-2025
Altair® GateVision PRO™ ©2002-2025
Altair® Geomechanics Director™ ©2011-2022
Altair® HyperCrash® ©2001-2023
Altair® HyperGraph® ©1995-2025
Altair® HyperLife® ©1990-2025
Altair® HyperMesh® ©1990-2025
Altair® HyperMesh® CFD ©1990-2025
Altair® HyperMesh® NVH ©1990-2025
Altair® HyperSpice™ ©2017-2025
Altair® HyperStudy® ©1999-2025
Altair® HyperView® ©1999-2025
Altair® HyperView Player® ©2022-2025
Altair® HyperWorks® ©1990-2025
Altair® HyperWorks® Design Explorer ©1990-2025
Altair® HyperXtrude® ©1999-2025
Altair® Impact Simulation Director™ ©2010-2022
Altair® Inspire™ ©2009-2025
Altair® Inspire™ Cast ©2011-2025
Altair® Inspire™ Extrude Metal ©1996-2025

Altair® Inspire™ Extrude Polymer ©1996-2025
Altair® Inspire™ Form ©1998-2025
Altair® Inspire™ Mold ©2009-2025
Altair® Inspire™ PolyFoam ©2009-2025
Altair® Inspire™ Print3D ©2021-2025
Altair® Inspire™ Render ©1993-2025
Altair® Inspire™ Studio ©1993-20245
Altair® Material Data Center™ ©2019-2025
Altair® Material Modeler™ ©2019-2025
Altair® Model Mesher Director™ ©2010-2025
Altair® MotionSolve® ©2002-2025
Altair® MotionView® ©1993-2025
Altair® Multi-Disciplinary Optimization Director™ ©2012-2025
Altair® Multiscale Designer® ©2011-2025
Altair® newFASANT™©2010-2020
Altair® nanoFluidX® ©2013-2025
Altair® NLView™ ©2018-2025
Altair® NVH Director™ ©2010-2025
Altair® NVH Full Vehicle™ ©2022-2025
Altair® NVH Standard™ ©2022-2025
Altair® OmniV™©2015-2025
Altair® OptiStruct® ©1996-2025
Altair® PhysicsAI™©2021-2025
Altair® PollEx™ ©2003-2025
Altair® PollEx™ for ECAD ©2003-2025
Altair® PSIM™ ©1994-2025
Altair® Pulse™ ©2020-2025
Altair® Radioss® ©1986-2025
Altair® romAI™ ©2022-2025
Altair® RTLvision PRO™ ©2002-2025
Altair® S-CALC™ ©1995-2025
Altair® S-CONCRETE™ ©1995-2025
Altair® S-FRAME® ©1995-2025

Altair® S-FOUNDATION™ ©1995-2025
Altair® S-LINE™ ©1995-2025
Altair® S-PAD™ © 1995-2025
Altair® S-STEEL™ ©1995-2025
Altair® S-TIMBER™ ©1995-2025
Altair® S-VIEW™ ©1995-2025
Altair® SEAM® ©1985-2025
Altair® shapeAI™©2021-2025
Altair® signalAI™©2020-2025
Altair® Silicon Debug Tools™©2018-2025
Altair® SimLab® ©2004-2025
Altair® SimLab® ST ©2019-2025
Altair® SimSolid® ©2015-2025
Altair® SpiceVision PRO™ ©2002-2025
Altair® Squeak and Rattle Director™ ©2012-2025
Altair® StarVision PRO™ ©2002-2025
Altair® Structural Office™ ©2022-2025
Altair® Sulis™©2018-2025
Altair®Twin Activate®©1989-2025
Altair® UDE™ ©2015-2025
Altair® ultraFluidX® ©2010-2025
Altair® Virtual Gauge Director™ ©2012-2025
Altair® Virtual Wind Tunnel™ ©2012-2025
Altair® Weight Analytics™ ©2013-2022
Altair® Weld Certification Director™ ©2014-2025
Altair® WinProp™ ©2000-2025
Altair® WRAP™ ©1998-2025

Altair HPCWorks®, a HPC & Cloud Platform
Altair® Allocator™ ©1995-2025
Altair® Access™ ©2008-2025
Altair® Accelerator™ ©1995-2025
Altair® Accelerator™ Plus ©1995-2025
Altair® Breeze™ ©2022-2025

Altair® Cassini™ ©2015-2025
Altair® Control™ ©2008-2025
Altair® Desktop Software Usage Analytics™ (DSUA) ©2022-2025
Altair® FlowTracer™ ©1995-2025
Altair® Grid Engine® ©2001, 2011-2025
Altair® InsightPro™ ©2023-2025
Altair® InsightPro™ for License Analytics ©2023-2025
Altair® Hero™ ©1995-2025
Altair® Liquid Scheduling™ ©2023-2025
Altair® Mistral™ ©2022-2025
Altair® Monitor™ ©1995-2025
Altair® NavOps® ©2022-2025
Altair® PBS Professional® ©1994-2025
Altair® PBS Works™ ©2022-2025
Altair® Simulation Cloud Suite (SCS) ©2024-2025
Altair® Software Asset Optimization (SAO) ©2007-2025
Altair® Unlimited™ ©2022-2025
Altair® Unlimited Data Analytics Appliance™ ©2022-2025
Altair® Unlimited Virtual Appliance™ ©2022-2025

Altair RapidMiner®, a Data Analytics & AI Platform
Altair® AI Hub ©2023-2025
Altair® AI Edge™ ©2023-2025
Altair® AI Cloud ©2022-2025
Altair® AI Studio ©2023-2025
Altair® Analytics Workbench™ ©2002-2025
Altair® Graph Lakehouse™ ©2013-2025
Altair® Graph Studio™ ©2007-2025
Altair® Knowledge Hub™ ©2017-2025
Altair® Knowledge Studio® ©1994-2025
Altair® Knowledge Studio® for Apache Spark ©1994-2025
Altair® Knowledge Seeker™ ©1994-2025
Altair® IoT Studio™ ©2002-2025
Altair® Monarch® ©1996-2025

Altair® Monarch® Classic ©1996-2025

Altair® Monarch® Complete™ ©1996-2025

Altair® Monarch® Data Prep Studio ©2015-2025Altair® Monarch Server™ ©1996-2025

Altair® Panopticon™ ©2004-2025

Altair® Panopticon™ BI ©2011-2025

Altair® SLC™ ©2002-2025

Altair® SLC Hub™ ©2002-2025

Altair® SmartWorks™ ©2002-2025

Altair® RapidMiner® ©2001-2025

Altair One® ©1994-2025

Altair® CoPilot™©2023-2025

Altair® Drive™©2023-2025

Altair® License Utility™ ©2010-2025

Altair® TheaRender® ©2010-2025

OpenMatrix™ ©2007-2025

OpenPBS® ©1994-2025

OpenRadioss™ ©1986-2025

Third Party Software Licenses

For a complete list of Altair Accelerator Third Party Software Licenses, please click [here](#).

Technical Support

Altair provides comprehensive software support via web FAQs, tutorials, training classes, telephone and e-mail.

Altair One Customer Portal

Altair One (<https://altairone.com/>) is Altair's customer portal giving you access to product downloads, Knowledge Base and customer support. We strongly recommend that all users create an Altair One account and use it as their primary means of requesting technical support.

Once your customer portal account is set up, you can directly get to your support page via this link: www.altair.com/customer-support/.

Altair Training Classes

Altair training courses provide a hands-on introduction to our products, focusing on overall functionality. Courses are conducted at our main and regional offices or at your facility. If you are interested in training at your facility, please contact your account manager for more details. If you do not know who your account manager is, e-mail your local support office and your account manager will contact you

Index

A

activate overbooking [38](#)
Allocator and TIMEVAR [36](#)
Allocator procedures [13](#)
Allocator procedures backward compatibility [27](#)
Altair Allocator Administrators Guide [3](#), [4](#)

C

command shell output [48](#)
configure a failover server replacement [49](#)
configure Allocator [9](#)
configure an LSF or OpeLava elim [61](#)
configure an LSF or OpenLava queue [60](#)

D

debugging [42](#)

F

failover configuration, tips [50](#)
failover server candidates [49](#)
failover.sh [50](#)

H

how vovserver failover works [50](#)

L

license overbooking [38](#)

M

map host names [33](#)
multiple queues [53](#)

O

overbooking advantages [38](#)
overbooking in Allocator [40](#)
overbooking operation [38](#)

P

passive mode [37](#)
PBS and other queues [58](#)

Q

queues example [54](#)

R

resource groups [35](#)

S

server candidates [49](#)

servercandidates.tcl [49](#), [50](#)

serverinfo.tcl [50](#)

setup.tcl [50](#)

sharing licenses across multiple sites [8](#)

standalone user interface [47](#)

start, stop and monitor Allocator [6](#)

T

troubleshooting elmins [62](#)

U

usage examples [29](#)

V

view allocation details [43](#)

vovautostart [49](#)

vovproject [50](#)

vovserverdir [49](#)

vovservsel.tcl [50](#)

vovtasker [50](#)

vovtaskers [49](#)

W

web interface [43](#)