



ALTAIR

ONLY FORWARD

VOV Subsystem 2025.1.2

Reference Guide

Contents

VOV Subsystem Reference Guide	7
Command Line Interface	8
CLI Commands	8
VOV Properties	10
Customization of Node.cgi	15
vovshowconnection	16
vovwavepp	17
vovmemtime	18
vovtestsocket	18
Fields, Selection Rules and Formatting Strings	20
Node Fields	20
Selection Rules	33
Selection Rules Operators	37
Formatting Strings	39
Failure Codes	40
VOV Metrics	42
Metrics Storage	42
Metrics Naming	42
Built-in Server Metrics	43
VOV Metrics Command Line Interface	43
Using vovselect	44
Vtk Interface	45
Plotting	45
Miscellaneous	46
Inner Loop Timers	46
Sequence of Firing of a Job on a Tasker	47
Error Message: "Too many notify clients for user ..."	48
Error message: "Cannot accept more connections"	49
vtk_server_config	49
Expected Duration	54
Environment Variables	55
Other Files Used by VOV	97
HTML Pages	98
Raw HTTP/VOV Interface	98
VOV Security Infrastructure	102
VOV Security Keys	102
vovsecurity	108
Client User Authentication	110
REST Application Programming Interface	112
Key REST Concepts	112
Resource Path URLs	113

Get Ready for REST.....	115
Example Application: REST 101.....	116
Example Application: REST 101 Use Key Authentication.....	119
Example Application: Job Submit and List.....	120
Launch Jobs with Non-default Options.....	120
The vov_rest_v3.py Python Library Module.....	121
REST Request Detailed Documentation.....	122
Issue REST Requests from the Swagger Web UI.....	123
Advanced REST Usage.....	125
JWT Token Allocation.....	128
All About Tcl.....	130
Tcl Syntax.....	130
Tcl Primitives.....	134
Tcl VOV Extensions.....	142
Tcl Procedures (Alphabetical).....	143
VTK API Available in vovtasker.....	227
VTK Procedures.....	229
vtk_acl.....	229
vtk_ae_getkeysfromfile.....	230
vtk_alert.....	230
vtk_annotation.....	231
vtk_artificial.....	233
vtk_asynch.....	233
vtk_bar.....	234
vtk_base64.....	234
vtk_cache.....	235
vtk_chain.....	235
vtk_checkout.....	236
vtk_checkouts.....	237
vtk_client.....	238
vtk_clients.....	239
vtk_cmd.....	240
vtk_crc.....	240
vtk_dailylog.....	240
vtk_date.....	242
vtk_debug.....	242
vtk_disable.....	243
vtk_equivalence.....	244
vtk_equivalences.....	245
vtk_event.....	247
vtk_eventfilter.....	250
vtk_eventmonitor.....	251
vtk_eventwidget.....	251
vtk_exclude.....	251
vtk_exit.....	252
vtk_feature.....	252
vtk_features.....	255

vtk_file.....	255
vtk_filesystem.....	256
vtk_flexlm.....	256
vtk_foreign_bucket_reset.....	258
vtk_fs.....	258
vtk_fsgroup.....	259
vtk_ftlm.....	261
vtk_generate_ae_keypair.....	262
vtk_generic.....	262
vtk_geo.....	263
vtk_get.....	263
vtk_graph.....	264
vtk_groupname.....	265
vtk_gui.....	265
vtk_gzip.....	266
vtk_hash.....	266
vtk_host.....	266
vtk_hostname.....	267
vtk_init.....	268
vtk_input.....	268
vtk_integer.....	269
vtk_job.....	270
vtk_jobclass.....	271
vtk_jobqueue.....	273
vtk_la_ctrl.....	274
vtk_licdaemon.....	274
vtk_licdaemons.....	276
vtk_license.....	276
vtk_licmon.....	277
vtk_limits.....	277
vtk_logname.....	278
vtk_lsf.....	279
vtk_metric.....	279
vtk_microcode.....	280
vtk_modify_fields.....	280
vtk_mq_enable_priority_based_alloc.....	281
vtk_mq_get_priority_based_alloc_percentage.....	281
vtk_mq_set_priority_based_alloc_percentage.....	281
vtk_mqsite_create_or_modify.....	282
vtk_multiqueue.....	282
vtk_new_process_session.....	284
vtk_node.....	284
vtk_nodeeditor.....	290
vtk_nodes.....	290
vtk_nodestatus.....	291
vtk_nodeviewer.....	292
vtk_object.....	292

vtk_output.....	293
vtk_path.....	295
vtk_percent.....	298
vtk_pie.....	298
vtk_ping.....	299
vtk_place.....	299
vtk_port.....	302
vtk_ppid.....	302
vtk_preemptrule.....	302
vtk_product.....	306
vtk_prop.....	306
vtk_protocol.....	308
vtk_pty_server_open.....	309
vtk_reservation.....	309
vtk_resource.....	312
vtk_resourcemap.....	312
vtk_resources.....	320
vtk_retrace.....	320
vtk_sanity.....	324
vtk_select.....	325
vtk_selectrule.....	328
vtk_server.....	329
vtk_set.....	331
vtk_setbrowser.....	339
vtk_sign.....	339
vtk_signal.....	340
vtk_strings.....	340
vtk_substitute.....	340
vtk_swd.....	341
vtk_tasker.....	341
vtk_taskerlist.....	350
vtk_taskerlists.....	350
vtk_taskerset.....	350
vtk_tcl.....	351
vtk_time.....	351
vtk_timestamp.....	352
vtk_trace.....	352
vtk_transition.....	355
vtk_umask.....	365
vtk_unixgroup.....	366
vtk_user.....	366
vtk_usergroup.....	369
vtk_utime.....	372
vtk_validate_keyformat.....	372
vtk_wave.....	372
vtk_wavegroup.....	384
vtk_waveplot.....	385

vtk_which.....	386
vtk_widget.....	387
vtk_write.....	387
Legal Notices	388
Intellectual Property Rights Notice.....	389
Technical Support.....	395
Index	396

VOV Subsystem Reference Guide

1

This chapter covers the following:

- [Command Line Interface](#) (p. 8)
- [Fields, Selection Rules and Formatting Strings](#) (p. 20)
- [VOV Metrics](#) (p. 42)
- [Miscellaneous](#) (p. 46)
- [VOV Security Infrastructure](#) (p. 102)
- [REST Application Programming Interface](#) (p. 112)
- [All About Tcl](#) (p. 130)
- [VTK Procedures](#) (p. 229)

Command Line Interface

CLI Commands

The Altair Accelerator products have a rich set of programs that can be run the shell command line. Each program performs a specific task and is controlled by a set of command line parameters. The programs provide a usage command when run with an `-h` parameter as a quick way to understand what each does and what parameters it takes.

CLI commands are case insensitive. For example, `timetolerance` and `timeTolerance` represent the same command. The keyword expressions AND, OR and NOT are also case insensitive.



Note: Resource expressions can be entered as lists in place of OR.

For example:

```
% nc run -r License:foo,User:abc -- sleep 10
```

Common CLI Commands

vovproject	Create, start, enable, stop a project.
vovconsole	Start the Graphical User Interface (GUI).
vovcheck	Run many checks on the setup of VOV.
vls	List the status of the files with respect to the dependency graph.
vsc	Show the c onsequences of changing one node.
vsh	Report the h istory of a file or a job.
vsi	Report basic the i nformation about the graph.
vsm	M onitor taskers, jobs, resources.
vsr	Issue a r etrace request.
vst	Show t ools executed in the current directory.
vsx	Show inputs and outputs of nodes.
vsy	Show why (Y) a node is not VALID.
vsz	Z ap away bad nodes.

Main User Commands

vovblast	Simultaneously remove and forget files
vovbrowser	Help start the browser interface by computing the port number for a project.
vovbuild	Build complex flows.
vovcleandir	Eliminate old log files that are no longer part of the flow.
vovcleanup	Eliminate old log files from a host, a project.
vovfind	Find file nodes in the graph.
vovfire	Execute a job "here and now".
vovforget	Forget objects in the in-memory database: nodes, sets, users, etc.
vovkill	Kill a process and all of its children.
vovjobqueue	Show all jobs in the queue.
vovrename	Rename nodes in the graph.
vovset	Create, edit, forget sets.
vovsh	The main Altair Accelerator client, with extended Tcl interpreter.
vovshow	Show things: users, buckets, nodes, sets, etc.
vovstop	Stop things: jobs, projects, retraces, ...
vovtouch	Toggle status of nodes.
vovversion	Show the version of the current installation.

Administration Commands

vovarch	Print architecture of current host.
vovenv	Useful to build environments.
vovequiv	Check equivalence rules for file names.
vovinit.bat	Setup the Windows NT shell.
vovremove	Remove files intelligently.
vovresourced	Read the <code>resources.tcl</code> file.
vovserver	The main vovserver.

vovserverdir	Show the working directory of the server.
vovservermgr	Modify settings in the running vovserver.
vovsetupuser	Setup the user account (UNIX only).
vovtasker	The main executable for a vovtaskers.
vovtaskermgr	Start and stop taskers.
vovtaskerroot	A tasker that can switch user ID (UNIX only).

Other Commands

vovshowconnection	Show connection path between two nodes in the graph.
vovcheckfiles	Force a check of all timestamps.
vovflowcompiler	Compile a flow to be included into the Flow Library.
vovprop	Add, modify and delete properties on files and jobs.
vovrestoretrace	Restore a corrupted graph either from a backup or from a saved copy.
vovwait4server	Wait for the server to be operative.
vovxrhs	Open a new <code>xterm</code> with the <code>DISPLAY</code> variable properly set (UNIX only).

VOV Properties

Properties can be attached to any object in the vovserver. Properties are managed from the command line via the `vovprop` command. This command can get, set, delete, and show properties on objects.

The name of the property must be less than 256 characters long and may contain alphanumeric characters and underscore (`_`), dash (`-`), and dot (`.`).

For string properties, the value must be less than 32,768 characters long and may contain any character.

```
vovprop: Usage Message
```

```
USAGE:
```

```
% vovprop SET [-text] objectIdList propName propValue
% vovprop GET [-default DEFAULT_VALUE_IF_PROP_MISSING] objectId propName
% vovprop DELETE objectId propName
% vovprop SHOW objectId [-namepat regexp] [-nameonly]
% vovprop SHOW [-namepat regexp] [-nameonly] id1 [id2 ... idN]
```

```
The action (GET, SET, DELETE, DEL, SHOW, LIST) is case-insensitive.
```

LIST and SHOW are equivalent.

OPTIONS FOR GET:

-default V -- Specify a default value in case objectId
or property is missing.

OPTIONS FOR SET:

-integer -- Specify that property is of type integer.
-text -- Specify that property is of type string.
-auto -- Determine type automatically (default).
-sticky -- The property is sticky (default).
-nosticky -- The property is not sticky.
-asynch -- Do not wait for reply from server.
-synch -- Wait for reply from server (default).
-elements -- If setting a property on a set, do so for the
elements contained in the set instead of the
set itself.
-noelements -- If setting a property on a set, do so on the set
itself (default).

OPTIONS FOR SHOW,LIST:

-namepat R -- Show only properties where name matches regexp R
-nameonly -- Show only name, and omit value

OTHER OPTIONS:

-v -- Increase verbosity.
-h -- Print this help.

EXAMPLES:

```
% vovprop SET -text 1 "ABC" "xyz"
% vovprop SET -text 1 "NUM" 1234
% vovprop GET 000123456 ABC
% vovprop GET -default "" 000123456 ABC
% vovprop DEL 000123456 ABC
% vovprop SHOW 000123456
```

NOTES:

A property of a job that is not 'sticky' will be forgotten
if the job is re-run successfully.

Properties can also be managed using the Tcl API's [vtk_prop_get](#) and [vtk_prop_set](#). Use [vtk_prop_list](#) to get a list of the names of the properties attached to an object.

Protected Properties

The properties "CAPSULE", "PRECMD" and "POSTCMD", which imply the execution of some code, are considered protected, because when they are attached to a job, they can only be modified by the owner of the job. Not even an ADMIN user can change them. This is consistent with the fact that only the owner of a job can change its main characteristics, namely the command line, the environment, or the working directory.

Common Properties

Property Name	Description
ALLPIDS	The ALLPIDS property holds a list of process IDs (PIDs) for all extant processes associated with the job. The process list includes: the subtasker process ("vovtaskerroot" or "vovtasker"),

Property Name	Description
	the wrapper (normally "vw"), the user specified job command and all active child/grandchild processes of these processes as determined by the processes' parent process ID (PPID) links. Then all active processes with the same process group id (PGID) or session id (SID), as any of the these process are added. Then all processes with the VOV_JOBID and VOV_PROJECT_NAME environment variables set for the current job and project are added. Known parent processes of the subtasker are excluded for each of these categories.
CAPSULE	The value is a Tcl script that is executed as a capsule on the fly. This is a protected property, only the job owner can modify it.
CHANGEGRAB	The value is a documentation of all changes in the grabbed resources list. The value is a list consisting of repeated "timestamp resourcename quantity" triplets for each change event. The property is not used anywhere else.
EXTLINKS	Attached to "trace" has list of HREF links to show in top part of browser interface pages.
JOB_SKIP_REASON	Set by <code>vovjobskip</code> when the skip flag is set.
MAILTO	Used with jobs submitted with option -m or -M
PIPELOG	Used with jobs submitted with option -wl
PRECMD	The command to be executed before a job is launched. If the command exits with a status different from 0 (zero), the job will not be launched. This is a protected property, only the job owner can modify it.
POSTCMD	The command to be executed after a job has completed, regardless of success or failure. This is a protected property, only the job owner can modify it.
PREEMPTED	Used in preemption.
PREEMPTRESUME	Used in preemption. The date of the most recent time a job was resumed.
PREEMPTTIME	Used in preemption.
PRODUCT	Attached to the "Trace" object (VovId=1) holds the internal name of the product implemented by the server. Typical values are "lm" and "lms" and "nc". The value is particularly important when starting a Monitor project while RLM is down.
PTYSERVER	Used with jobs submitted with option -Ir and -Il.

Property Name	Description
RESUMEARGS	Used in preemption, in case of manual resumption.
RESUMEDELAY	Used in preemption, used by <code>vovjobresumer</code> .
RESUMERJOBID	Used in preemption to hold the ID of the resumer job for a preempted job.
SOLUTION	The list of tasker-local resources for a job. These resources are criteria for tasker selection when a job is scheduled.
TRIGGER	Names a trigger to be called by <code>vovtriggerd</code> .
WHY	Explain why jobs have the status they have.
FlowTracer Properties	
FORCE_VALIDATE	Used by the Force Validate technique to annotate jobs that have been forcefully validated.
Accelerator Properties	
DONOTLMREMOVE	This is an optional property for a job. If set, this variable disables the automatic cleanup of license handles matched to a suspended job. This cleanup is performed by <code>vovresourced</code> .
NC_DEFAULT_JOBCLASS	This is an optional property of the trace (<code>VovId=1</code>). If it is set, it indicates the name of the jobclass to be used by default. You can use <code>vovprop</code> to set this property. Example: <pre>% nc cmd vovprop SET -text 1 "NC_DEFAULT_JOBCLASS" "normal"</pre>
NC_VALID_DIRECTORIES	This is an optional property of the trace (<code>VovId=1</code>). If it is set, it indicates a list of prefixes for directories that should be accepted as valid by <code>'nc run'</code> . This is in place of using the <code>-D</code> option in <code>'nc run'</code> . The following example accepts <code>'.'</code> as a valid directory. <pre>% nc cmd vovprop SET -text 1 "NC_VALID_DIRECTORIES" "."</pre>
IDLE_WARN_DELAY	This is an optional property of the set representing the job class, which is the set <code>"Class:CLASSNAME"</code> . This property is set with the procedure <code>vtk_jobclass_set_idle_delays</code> and is used by <code>vovnotifyd</code> .
IDLE_KILL_DELAY	This is an optional property of the set representing the job class, which is the set <code>"Class:CLASSNAME"</code> . This property is set with

the procedure `vtk_jobclass_set_idle_delays` and is used by `vovnotifyd`.

AGGRESSIVE_SCHEDULING_DELAY is an optional property of the set representing the job class, which is the set "Class:CLASSNAME". It represent the time after which the system automatically revokes resources that have been grabbed but are not yet used. This property is created by the script `jobclass.cgi` or with the procedure `vtk_jobclass_set_revocation_delay`. This is the old name of the property. The new variable is `REVOKE_DELAY`.

REVOKE_DELAY This is an optional property of the set representing the job class, which is the set "Class:CLASSNAME". It represent the time after which the system automatically revokes resources that have been grabbed but are not yet used. This property is created by the script `jobclass.cgi` or with the procedure `vtk_jobclass_set_revocation_delay`. This is the new name for a property previously called `AGGRESSIVE_SCHEDULING_DELAY`. The property can also be attached to a resource map.

MAX_RESCHEDULE This is an optional property of the set representing the job class, which is the set "Class:CLASSNAME". It represent the maximum number of times a job is to be rescheduled after a fast failure. This overrides the value of the global parameter `autoRescheduleCount`.

FT_LOGS Property created by utility 'vovrun', attached to the NC job, used to store a list of logs of the FT job. This property is used by "`nc info -ft XXX`"

NCJOBID Automatic property. Attached to a FlowTracer job that is being executed via NC using an indirect tasker. It contains the VovId of the corresponding job in NC.

NC_JOBID Automatic property. Redundant with NCJOBID. **Do not use.** This property is set by the script `vovrun` and will be eliminated in future releases.

NCSETNAME Optional property that may be attached to a FlowTracer job. If this property is set, and the job is executed on NC using an indirect tasker, the property value indicates the name of a set to be used in the NC context to contain the NC job corresponding to the FlowTracer job. This property is used in the file `$VOVDIR/etc/tasker_scripts/taskerVNC.tcl`.


Monitor Properties

MYVIEW Attached to the each VovUser object, specifies the pair "owner view" of a view.

LM_LAST_LIVE_LOAD_TS	Attached to the trace (VovId=1). Used by <code>live_load_checkouts.tcl</code>
LM_FORECAST_ACCOUNT_TYPE	Attached to the trace (VovId=1). Used by the <code>ftlm_forecast</code> .
LM_FORECAST_SIMPLE_LOCK	Attached to the trace (VovId=1). Used to lock/unlock the forecast database
LM_FORECAST_NOTIFY_TS	Attached to the trace (VovId=1). Used to store the time of the last forecast notification.
LM_FORECAST_DONE_userName	Attached to the trace (VovId=1).
LM_FORECAST_START_TS, LM_FORECAST_FINISH_TS	Attached to the trace (VovId=1). Used to store the interval of forecast.

Customization of Node.cgi

The `node.cgi` script can be customized by means of these properties:

DESCRIPTION	This property must be attached to the node to be displayed. It consists of a string that is displayed in the Job Description section of the page. This property may go away because it is redundant with the job name field.
FIRENOW_RESOURCES	<p>If this property is defined in object with <code>VovId=1</code> (the <code>VovTrace</code>), it represents a resource expression for a job. This property activates the "FireNow!" capability in <code>node.cgi</code>. The  icons appears in the top right corner and if clicked, it causes the resources for the job to be replaced with</p> <pre>(@FIRENOW_RESOURCES@) OR (@ORIGINAL_RESOURCES@)</pre> <p>and the job is rescheduled at top priority. This assumes that there are taskers that offer the resource "<code>@FIRENOW_RESOURCES@</code>". A typical example of use of this property is</p> <pre>% vovprop set 1 FIRENOW_RESOURCES "fireNow"</pre>
NODE_CGI_OUTPUT_SUMMARY	This property must be attached to the trace object (VovId=1). It consists of a space-separated list of glob expressions. If an output file matches the glob expression, it will be displayed in the page. (See also <code>OUTPUT_SUMMARY</code> below.)

OUTPUT_SUMMARY

This property must be attached to the node to be displayed. It consists of a space-separated list of glob expressions. If an output file matches the glob expression, it will be displayed in the page.

vovshowconnection

Show connection between two nodes in the trace.

```
vovshowconnection: Usage Message

DESCRIPTION:
  Show connection between two nodes in the trace.

USAGE:
  % vovshowconnection \[option\] -from NODESPEC -to NODESPEC

  where NODESPEC is either a VovId or the name of a file.

OPTIONS:
  -l          -- longer format without label field size limit
  -ll         -- full    format without any field size limits
  -O <format> -- choose output format. For reference:
                default = "@LEVEL:4@ @NODETYPE:6@ @ID@ @LABEL:70@"
                long    = "@LEVEL:3@ @NODETYPE:6@ @ID@ @LABEL@"
                longer   = "@LEVEL@ @NODETYPE@ @ID@ @LABEL@"

EXAMPLES:
  % vovshowconnection -from 00123456 -to 00223344
  % vovshowconnection -ll -from aa -to zzz
  % vovshowconnection -O "@LEVEL@ @ID@ @NAME@" -from aa -to zzz
```

Comments

The command works by using the fast, powerful, set operations in FlowTracer to find the down-cone of the -from node, and the up-cone of the -to node, and then finding their intersection.

The command has two mandatory parameters, the 'from' nodespec and the 'to' nodespec.

```
-from  Specify the nodespec of the 'from' node
-to    Specify the nodespec of the 'to' node
```

A node specification, or 'nodespec' for short, is either the VovID of a place (file) or a transition (job), or the name of a file in the flow graph.

There are two options to control the output format, and one to get usage information.

```
-h  Show brief usage information
-l  longer format without label field size limit
-ll full format without any field size limits
```


vovwavepp

The utility `vovwavepp` is used to convert the verbose logs generated by the `vovserver` into efficient "wave" files.

This utility is generally called automatically by the plotting scripts (such as `taskerload.cgi`). There are several types of log files that can be processed:

- resources logs, in `*.swd/logs/resources*`, which contain information about the utilization of the resource maps
- taskerload logs, in `*.swd/logs/taskerload/*`, which contain information about average load, free RAM, and idle CPU times
- waitreasons logs, in `*.swd/logs/waitreasons/*`, which contain information about the reasons that prevent jobs from being dispatched

In general, the directory "waves" can be always blasted away, because they are recreated automatically the next time a plot is requested. If you would like to run `vovwavepp` from the command line, here are some examples.

Waves for Resources

To process all resources in the year 2009, you can run:

```
% cd `vovserverdir -p .`  
% vovwavepp -p resources -y 2009 -D logs -P resources.@DATE@.log
```

To restrict the processing to the month of June 2019, you can add the option `-m 6`:

```
% cd `vovserverdir -p .`  
% vovwavepp -p resources -y 2019 -m 6 -D logs -P resources.@DATE@.log
```

If you only want one resource, for example, `License:calibre`, add the option `-r RESOURCE`, as in

```
% cd `vovserverdir -p .`  
% vovwavepp -p resources -y 2009 -m 06 -D logs -P resources.@DATE@.log -r  
License:calibre
```

Waves for Tasker Load

To process all tasker load information in the month of June 2019, you can run:

```
% cd `vovserverdir -p logs`  
% vovwavepp -p taskerload -y 2019 -m 06 -D taskerload -P @DATE@
```

vovmemtime

vovmemtime is a utility to create simulated workloads. When this program runs, it occupies a specified amount of memory and creates high CPU utilization for the requested amount of time, to simulate the activity of real design programs running.

Usage

```
This program is used to test VOV with a high load
in terms of both CPU and memory utilization.
Usage: vovmemtime <mem_in_MB> <time_in_seconds> <recursion_levels>
      If time is negative, it represents the number of iterations
      vovmemtime 2000 120 0: is one cpu process (0 recursion) running for 2mn (120s)
      and walking a 2GB (2000 MB) memory space
      vovmemtime always use only one cpu at a time, no matter how many recursions
```

When the third argument is greater than 0, **vovmemtime** recursively starts instances of itself with the same memory and duration parameters. This is normally used in testing, to verify that the memory used by child processes is correctly tracked.

The CPU load is generated by a tight loop over the allocated memory, and does not include any I/O.

Examples

This shows an example of **vovmemtime** output, run on Windows.

```
vnc@comet [DEFAULT] X:\>vovmemtime 10 10 3
PID=781      Allocated 10 MB
PID=781      Executing 'vovmemtime 10 10 2'

PID=722      Allocated 10 MB
PID=722      Executing 'vovmemtime 10 10 1'

PID=786      Allocated 10 MB
PID=786      Executing 'vovmemtime 10 10 0'

PID=699      Allocated 10 MB
PID=699      Performed 219 loops on 2621440 elements
PID=786      Performed 200 loops on 2621440 elements
PID=722      Performed 219 loops on 2621440 elements
PID=781      Performed 215 loops on 2621440 elements
```

vovtestsocket

Test vovserver socket connectivity.

The command **vovtestsocket** attempts to connect to the specified port on a host. It must be called from a vov-project enabled shell. It prints the result and also places it in a property on the trace (object 1) in vovserver.



Important: The vovserver for the project must be running.

The result is printed and a time-stamped property of the form SOCKET_<HOST>_<PORT> is attached. See example below.

```
vovtestsocket: Usage Message

DESCRIPTION:
  This utility tests the connectivity to a socket
  and leaves the status in a property of the trace.
  The property is called SOCKET_${HOST}_${PORT}.

USAGE:
  % vovtestsocket HOST PORT
```

Examples

```
myproject@server DEFAULT ~ > vovtestsocket tiger 110
vovtestsocket 04/24/2017 10:21:55: message: Status: 'ok'
myproject@server BASE+SVN1 ~ > vovprop list 1 | grep SOCK
SOCKET_tiger_110 = '1493054515 ok'
```

This example attempts to connect to a host 'tiger' on port 110 (the one usually used by POP3 protocol). This command returns 0 exit status irrespective of the result.

Fields, Selection Rules and Formatting Strings

Node Fields

Each node in the trace has *field* attributes.

There are three types of fields:

1. Boolean fields, which take value in the set (0,1)
2. Integer fields, which take any 32-bit signed integer value
3. String fields, which take a null-terminated string value.

The fields are used in the [Selection Rules](#) and in [Formatting Strings](#). The field names are not case-sensitive. For example, the rules `IsJob` and `isjob` are equivalent.

To get an current list of all fields, use the following:

```
% vovshow -fields
```

Incompatible Fields

Some fields make sense only for one type of nodes. For example, the field `NAME` makes sense for files but not for jobs, while the field `COMMAND` makes sense for jobs but not for files. A field is considered incompatible for a node if it does not make sense for that node. This notion of incompatible fields is important in the construction of selection rules and formatting strings.

Fields List

Field Name	Field Type	Applies To	Description
AGE	Integer (duration)	Nodes	Age in seconds of a job or of a file.
AGEPP	String	Nodes	Pretty-printed version of age of a job or of a file.
ANNOTATIONS	Integer	Nodes	Number of annotations attached to the node.
AUTOKILL	Integer (duration)	Jobs	Time before a job is automatically killed.
AUXRESOURCES	String	Jobs	Additional resources assigned to a job by the server.
BUCKETID	String	Jobs	The internal ID of the bucket object that contains the specified job.

Field Name	Field Type	Applies To	Description
CHOSENTASKERID	String	Jobs	If a job must go to a specific tasker, the ID of that tasker.
COMMAND	String	Jobs	The command line for the job.
COMMANDLENGTH	Integer (size)	Jobs	The length of the command line for the job.
CPUPROGRESS	Integer (percent)	Jobs	The percent of CPU time used by a jobs, including all its children, in the last sampling interval. This number can be greater than 100 if the job is running on a multi CPU machine. If CPUPROGRESS is zero, the job is stuck, probably waiting for input or for a license.
CPUTIME	Integer (milliseconds)	Jobs	The CPU time used by the job, in milliseconds, including all its children. This is a 64-bit number.
CURRAM	Integer (MB)	Jobs	The current amount of RAM used by the job, in MB
CURREAD	Integer (bytes)	Jobs	The current number of input I/O events for the job, in bytes.
CURVM	Integer	Jobs	The current amount of virtual memory used by the job, in MB
CURWRITER	Integer (bytes)	Jobs	The current number of output I/O events for the job, in bytes.
CWD	String	Jobs	The current working directory for the job.
DB	String	Files	The database of a file.
DIR	String	Nodes	Same as CWD.
DISPATCHDATE	Integer (timestamp)	Jobs	The time the job has been dispatched to a tasker. This is relevant for indirect taskers, where the DISPATCHDATE may be significantly different from the START time due to latency in the secondary queue.
DURATION	Integer (duration)	Jobs	The duration of a job, in seconds.
DURATIONPP	String	Jobs	The duration of a job ID in pretty format.
ENDDATE	String	Jobs	The end date of a job in string format.

Field Name	Field Type	Applies To	Description
ENDED	Integer (timestamp)	Jobs	The end date of a job in integer format. If the job is still running or retracing, this field has the value 0 (see Ended2 for a different value).
ENDED2	Integer (timestamp)	Jobs	The end date of a job in integer format. If the job is running or retracing, this field returns the current time stamp (see ENDED for a different value)
ENV	String	Jobs	The environment of a job.
ENVARGS	String	Jobs	From the environment string, all the words after the first.
EXECHOST	String	Jobs	The execution host of the job.
EXITSTATUS	Integer	Jobs	The exit status of a job.
EXPDUR	Integer (duration)	Jobs	(OBSOLETE! Use XDUR instead) Expected duration of a job, in seconds
EXPDURPP	String	Jobs	(OBSOLETE! Use XDURPP instead) Expected duration, pretty printed.
FAILCODE	Integer	Jobs	A mask of values indicating why a job failed.
FLAGS	String	Nodes	Obsolete. Not supported any longer.
FSEXCESS	Integer	Jobs	This is essentially (FSHISTORY - FSTARGET) + (FSRUNNING - FSTARGET). See also FSEXCESSRUNNING and FSEXCESSHISTORY
FSEXCESSHISTORY	Integer	Jobs	This is essentially (FSHISTORY - FSTARGET). See also FSEXCESS.
FSEXCESSRUNNING	Integer	Jobs	This is essentially (FSRUNNING - FSTARGET). See also FSEXCESS. If the number is positive, that means that the group to which the job belongs has more running jobs than it should.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	VovPreemptRule -rulename SOMENAME \ ... -preempting "FSEXCESSRUNNINGLOCAL<0" \ -preemptable "FSEXCESSRUNNINGLOCAL>0 JOBCLASS==@JOBCLASS@FSRANK9>@FSRANK9@" \ ...

Field Name	Field Type	Applies To	Description
FSEXCESSRUNNINGLOCA	Integer	Jobs	Similar to FSEXCESSRUNNINGLOCAL but uses the balance of running jobs at one level above the local.
FSEXCESSRUNNINGLOCA	Integer	Jobs	Similar to FSEXCESSRUNNINGLOCAL but uses the balance of running jobs at two levels above the local.
FSEXCESSRUNNINGLOCA	Integer	Jobs	Similar to FSEXCESSRUNNINGLOCAL but uses the balance of running jobs at three levels above the local.
FSEXCESSRUNNINGLOCA	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL.
FSEXCESSRUNNINGLOCA	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL1.
FSEXCESSRUNNINGLOCA	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL2.
FSEXCESSRUNNINGLOCA	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL3.
FSGROUP	Integer	Jobs	The FairShare group that the job belongs to.
FSHISTORY	Integer	Jobs	The actual share in the FairShare window for the FairShare group, multiplied by 10,000.
FSRANK	Integer	Jobs	The FairShare rank of the FairShare group to which the job belongs, or -1 if the FairShare group has no rank. (See also FSRANK9)
FSRANK9	Integer	Jobs	This is the same as FSRANK, except that the value returned for groups that have no FairShare rank is 9,999,999 instead of -1.
FSRUNNING	Integer	Jobs	The actual share of all running jobs for the FairShare group, multiplied by 10,000.
FSRUNNINGCOUNT	Integer	Jobs	The current number of running jobs for the FairShare group of the job.
FSSUBGROUP	String	Jobs	The part of the FairShare group after the colon.
FSSUSPENDEDCount	Integer	Jobs	The current number of suspended jobs for the FairShare group of the job.

Field Name	Field Type	Applies To	Description
FSTARGET	Integer	Jobs	The FairShare target for the FairShare group to which the job belongs. The target, which is normally a fractional number less than 1.0, is multiplied by 10,000 to yield the FSTARGET. For example, a group that has a target of 30%=0.3 will have a FSTARGET of 3,000.
FSTOKENS	Integer	Jobs	An integer multiplier for the contribution of the job to the FairShare (e.g. a fstokens value of 2 means that the job counts as 2 "normal" jobs)
FSUSER	String	Jobs	The user component of a FairShare node (i.e. the component right after the dot '.', if it exists)
GRABBEDRESOURCES	String	Jobs	The list of resource maps that have been grabbed in order to dispatch a job to a tasker. This is valid only for RUNNING and RETRACING jobs, and the value of the field may change over time due to reconciliation of resources. See also the GRABBEDRESOURCESO field, which is available for jobs even after completion.
GRABBEDRESOURCESO	String	Jobs	The original list of grabbed resources when a job was last dispatched to a tasker. For RUNNING and RETRACING jobs it is best to look at GRABBEDRESOURCES instead. The final letter is an "oh" for Original.
GRABBEDTOOLS	String	Jobs	Obsolete: Only for running jobs
GROUP	String	Jobs	The FairShare group of a job. This is different from OSGROUP, which is used to specify the group permissions for the job.
HASANNOTATIONS	Boolean	NODES	True if the node has annotations.
HASINPUTCONFLICT	Boolean	JOBS	True if job failed on an input conflict.
HASINPUTS	Boolean	NODES	True if a node has one or more inputs.
HASOUTPUTCONFLICT	Boolean	JOBS	True if job failed on an output .conflict
HASOUTPUTS	Boolean	NODES	True if a node has one or more outputs.
HASPROFILE	Boolean	JOBS	True if job collects profile information like RAM and CPU usage at runtime (see option -profile in <code>nc run</code>)

Field Name	Field Type	Applies To	Description
HASRUNINFO	Boolean	JOBS	Obsolete: always 0.
HOST	String	JOBS	The host that executed a job.
ID	String	NODES	The ID of a node. This field is of type "string" and is shown with leading zeroes (e.g. "000123456"). Contrast this with the field IDINT which is of type "integer".
IDINT	Integer	NODES	The integer version of the Vovid of a node. Contrast this with the field ID which is of type "string"
INPUTS	Integer	NODES	The number of inputs of a node.
ISAUTOFLOW	Boolean	JOBS	The job turns VALID as soon as all its inputs are VALID, with no execution.
ISAUTOFORGET	Boolean	JOBS	The job is automatically forgotten after a certain time after completion.
ISAUTOKILL	Boolean	JOBS	The job is automatically killed if it exceeds its expected duration.
ISBARRIER	Boolean	NODES	True if a file has a barriers on it.
ISBARRIERINVALID	Boolean	NODES	True if a file has a barriers would be INVALID were it not for the presence of a barrier.
ISCHAIN	Boolean	FILES	True if the file is part of a chain, meaning that there are tools that operate 'in-place' on the file.
ISDATA	Boolean	NODES	True if node is a file.
ISEPHEMERAL	Boolean	JOBS	(experimental, do not use).
ISFILE	Boolean	NODES	True if node is a file.
ISINTERACTIVE	String	JOBS	True if the job is interactive (-I -Ir -Il -wl)
ISJOB	Boolean	NODES	True if node is a job.
ISMIGRATABLE	Boolean	JOBS	This field is an annotation to identify jobs that can be migrated. Currently not supported.
ISNODE	Boolean	NODES	True if a node is a node (Always true)
ISNONEXEC	Boolean	JOBS	True if a job is not executable

Field Name	Field Type	Applies To	Description
ISNONEXEC	Boolean	JOBS	True if the job is non-executable.
ISOPTIONAL	Boolean	FILES	True if an output has the OPTIONAL flag
ISPLACE	Boolean	NODES	OBSOLETE: True if a node is a file (see ISFILE)
ISPREEMPTABLE	Boolean	JOBS	This field is honored by the <code>vovpreemptd</code> daemon. If the value is zero, then the job is not considered for preemption. For information on how to set the preemptable flag, see <i>Control Whether a Job is Preemptable</i> . See also ISMIGRATABLE.
ISPROCESS	Boolean	NODES	Same as ISJOB.
ISQUEUED	Boolean	JOBS	The job is ready to fire and in a bucket in the job queue.
ISREADYTOFIRE	Boolean	JOBS	All inputs of a job are VALID and the job is ready to fire.
ISSCHEDULED	Boolean	NODES	True if node is scheduled to be retraced.
ISSCHEDULEDBARRIERIN	Boolean	NODES	True if node is scheduled and had the barrier invalid flag set.
ISSHARED	Boolean	FILES	True if file is shared output.
ISSKIP	Boolean	JOBS	True if the job is skipped (also called ISAUTOFLOW).
ISSTDERR	Boolean	FILES	True if file is a stderr file.
ISSTDOUT	Boolean	FILES	True if file is a stdout file.
ISSUBJOB	Boolean	JOBS	True if the job is a subjob (e.g. partialtool, vovjobresumer).
ISSUSPENDED	Boolean	JOBS	True if the job or one of its children is suspended.
ISSYSTEMJOB	Boolean	JOBS	True if the job is a 'system job' like vovzip, or vovsh -s netinfo.
ISTOOL	Boolean	NODES	Same as ISJOB and ISPROCESS.
ISTRANSFER	Boolean	JOBS	True if the job is being transfered to another cluster.
ISTRANSITION	Boolean	NODES	True if node is job (see ISJOB)

Field Name	Field Type	Applies To	Description
ISTRIGGERRUN	Boolean	FILES	True if the file has the trigger,run flag, i.e. if a change in the file triggers a retrace of the downcone
ISTRIGGERSTOP	Boolean	FILES	True if the file has the trigger,stop flag, i.e. if a change in the file triggers a stop of the downcone, followed by a retrace of the downcone
ISUNSAFE	Boolean	JOBS	True if node is "unsafe"
ISZIPPABLE	Boolean	PLACES	True if the file can be automatically zipped.
ISZIPPED	Boolean	PLACES	True if the file is currently zipped.
IUO	String	NODES	One of the following characters: "." "i" "u" "o"
JOBCLASS	String	JOBS	The jobclass of a job
JOBID	String	JOBS	The ID (number) of a job (a field of PROCESSES). An integer but often with leading 0s.
JOBLOGDIR	String	JOBS	The directory to which the logfile of a job is written (if a logfile is specified on the command line). If no logfile is specified, this is the current directory of the job.
JOBNAME	String	JOBS	The job name of a job
JOBPROJ	String	JOBS	Name of the project that the job belongs to (same as project).
JPP	String	JOBS	Job placement policy.
LABEL	String	NODES	A short label for the node.
LASTCUPROGRESS	Integer (timestamp)	JOBS	The last time the system detected some cpu progress in the job.
LASTCUPROGRESSPP	String	JOBS	The pretty-print version of "\$NOW-LASTCUPROGRESS".
LEGALEXIT	String	JOBS	The legal exit allowed for a job to be considered VALID (same as OKSTATUS).
LEVEL	Integer	NODES	The level of a node.
LMHANDLESALL	String	JOBS	A detailed list of all handles that have been matched to this job. For each handle, the field shows: ResourceName#Tokens MatchType

Field Name	Field Type	Applies To	Description
			FLEXlmHandle License:a#1 best 22334 License:b#2 sure 12345
LMHANDLESNRU	String	JOBS	A detailed list of all the NRU (Not Requested / Used) handles that have been matched to this job. For each handle, the field shows: ResourceName#Tokens MatchType FLEXlmHandle License:a#1 nru 22334 License:b#2 nru 12345
LMRESOURCES	String	JOBS	These are the license resources (typically derived from FLEXlm features) that appear to be used by a job even if the job does not explicitly declare them. This field is used by <code>vovlmremove</code> to decide which features to remove. The field is updated based on the matching of free FLEXlm handles and running jobs. The field consists of an even number of words, with each pair consisting of a resource name and a boolean flag indicating if the resource is a "false-out-of-queue", i.e. the handle is not-requested but used. A handle is Example: License:a#2 0 License:b#1 1 This means that the job requested 2 tokens of License:a and is in fact using them. In addition, the job is using 1 token of License:b even if it does not request it. See also LM_HANDLES_ALL and LM_HANDLES_NRU.
LX	String	JOBS	Legal Exit Value for a job (Same as <code>legalexit</code> and <code>ok_status</code>).
MAXRAM	Integer (MB)	JOBS	The max RAM used by a job and its children, in MB.
MAXSWAP	Integer	JOBS	Maximum amount of swap used by the job, in MB.
MAXVM	Integer (MB)	JOBS	Maximum amount of virtual memory used by the job, in MB.
NAME	String	FILES	The name of a file.
NAMEX	String	FILES	The name of a file, fully expanded. The expansion is done on the server side.
NODETIMESTAMP	Integer (timestamp)	NODES	The time the node last changed status (i.e. changes color).

Field Name	Field Type	Applies To	Description
NODETYPE	String	NODES	"FILE" if node is a file, "TOOL" if node is a job (I know, TOOL should be JOB)
NUMA	String	JOBS	Requested NUMA placement for the job.
OKSTATUS	String	JOBS	The list of acceptable exit status (same as LEGALEXIT).
OSGROUP	String	JOBS	The operating system group for the job (different from GROUP, which is used in the FairShare scheduling).
OUTPUTS	Integer	NODES	The number of outputs of a node.
PID	Integer	JOBS	The process id of a job.
PRIORITY	Integer	JOBS	Prioity of the job, in the range 0 to 15.
PRIORITYPP	String	JOBS	Pretty printed version of the priority of the job.
PROJECT	String	JOBS	Name of the project that the job belongs to (same as jobproj)
PROP.<propname>	String	NODES	This field is used to access the property with name <code>propname</code> . For example, to access the property "ABC", one should ask for "PROP.ABC". See also field PROPERTIES.
PROPERTIES	String	NODES	The list of properties attached to a node. The list consists of an even number of words, where the first word is either 'S' (for STRING) or 'I' for INTEGER properties and the second word is the name of the property. See also the 'PROP.*' field
QUEUETIME	Integer (timestamp)	JOBS	Time the job was put onto the job queue, more specifically the last time the job was scheduled. The job could in fact enter the job queue later due to dependency constraints or to lack of space in queue.
QUEUEWAIT	Integer (duration)	JOBS	The time a job has waited in the queue, in seconds.
QUEUEWAITPP	String	JOBS	Pretty-print version of QUEUEWAIT.
RANDOM	Integer	ALL	A random number in the range [0-9999]. Used, for example, to sort jobs in preemption rules.

Field Name	Field Type	Applies To	Description
REQCORES	Integer	JOBS	Requested CORES for the job. This is set when the job is dispatched to a tasker. Another name for this field is REQCPUS. See also explanation on REQRAM.
REQPERCENT	Integer	JOBS	Requested PERCENT for the job. See also explanation on REQRAM.
REQRAM	Integer (MB)	JOBS	RAM requested by a job. This value may come from 1) the SOLUTION property on the job, if it exists, 2) from the requested resource on the bucket, if the job is queued, or 2) from the resources string of the job. Normally, the value is the same regardless of the origin, but it is possible and acceptable that the value may change due to additional math performed during scheduling. Normal case: a job requests "RAM/200". Then the REQRAM value is going to be 200. Strange case: a job requests "RAM/200 SLOTS/2 RAM/300" (i.e. the RAM request is repeated). In this case, REQRAM will be 200 for the job if it is not Queued or if the SOLUTION property is removed after the execution of the job. Else it will be 500 (=200+300) while the job is Queued.
REQSLOTS	Integer	JOBS	Requested number of slots for the job. See also explanation on REQRAM.
REQSWAP	Integer (MB)	JOBS	Requested swap for the job. See also explanation on REQRAM.
RESERVEDRESOURCES	String	JOBS	DO NOT USE: Only for running jobs
RESERVEDTOOLS	String	JOBS	DO NOT USE: Only for running jobs
RESOURCES	String	JOBS	The resources of a job.
RETRACINGID	String	JOBS	The id of the retracing job (to find the other job is a RUNNING/RETRACING pair).
RUNSTATUS	String	JOBS	A representation of how well the job is running. Typical values are Good, Paging, NoCpu. Check <i>Job Runtime - Monitor and Profile</i> for more information.
SCHEDTIME	Integer (timestamp)	JOBS	The earliest time that this job can be scheduled (set with -at or -after option in nc run).

Field Name	Field Type	Applies To	Description
TASKERGROUP	String	JOBS	The tasker group of the tasker to which the job has been dispatched.
TASKERID	String	JOBS	The id of the tasker to which the job has been dispatched.
TASKERNAME	String	JOBS	The tasker name to which the job has been dispatched.
TASKERSLOTSSUSPENDA	Integer	JOBS	Number of suspendable jobs on the tasker on which the job is running.
TASKERSLOTSSUSPENDE	Integer	JOBS	Number of slots suspended on the tasker on which the job is running.
TASKERSLOTSUSED	Integer	JOBS	Number of slots used on the tasker on which the job is running.
STARTDATE	String	JOBS	Like STARTED, only in nice formatted ASCII
STARTED	Integer (timestamp)	JOBS	The time the job was started, or -1 if the job never ran.
STATUS	String	NODES	The status of the node.
STATUSCOLOR	String	NODES	The color associated to the node
STATUSCONSTRAST	String	NODES	A color that contrasts with STATUSCOLOR. Used mostly in templates for the browser interface.
STATUSIO	String	NODES	Short version of STATUS. For example, instead of VALID you get 'V' (the name is not very good)
STATUSMASK	Integer	NODES	(HARD TO USE) a binary mask with a bit set for each value of node status.
STATUSNC	String	NODES	job status for Accelerator.
STOLENRESOURCES	String	JOBS	These are the resources that have been taken away (revoked) from the job, typically because the job has been preempted.
SUBJOBIDS	String	JOBS	The ids of all subjobs of a job (used for example with distributed parallel jobs).
SUBMITHOST	String	JOBS	The submission host
SUBRESOURCES	String	JOBS	Subordinate resources for a job, i.e. the resources after the first '--' token. Used by indirect taskers.

Field Name	Field Type	Applies To	Description
SUSPENDEDINTERVALS	String	JOBS	The list of all intervals in which the job has been suspended.
SUSPENDEDTIME	Integer (duration)	JOBS	The total amount of time the job has been suspended (in seconds).
SUSPENSION	Integer (duration)	JOBS	The same as suspendtime.
TAIL	String	FILES	The name of the file not including the directory path.
TIMESTAMP	Integer (timestamp)	FILES	The time the file was last modified.
TIMESTAMPPP	String	FILES	Pretty-printed version of TIMESTAMP.
TOOL	String	JOBS	The name of the first tool used by the job.
TOPJOBID	String	JOBS	The id of the top job in a distributed parallel job.
USER	String	JOBS	The user (or owner) of a job.
USERXDUR	Integer (duration)	JOBS	The expected duration of a job as specified by the user. If negative, the expected duration has not been specified.
USERXDURPP	String	JOBS	The expected duration as specified by the user, pretty printed.
USESESSIONID	Boolean	JOBS	True if the vovtasker is supposed to look at the session id to find the processes related to this job (in addition to the parent-child relationship).
WAITREASON		JOBS	A description of why this job is waiting.
X	Integer	NODES	The X coordinate of a node.
XDUR	Integer (duration)	JOBS	The current expected duration of a job. It may be set by the user or calculated from the most recent successful completion of the job. If negative, the expected duration is not known.
XDURPP	String	JOBS	The current expected duration of a job, pretty printed.
XPRIORITY	Integer	JOBS	The execution priority of a job (range 0 to 15)
XPRIORITYPP	String	JOBS	Pretty-printed version of XPRIORITY.

Field Name	Field Type	Applies To	Description
Y	Integer	NODES	The Y coordinate of a node (same as LEVEL).
YOUNG	Boolean	FILES	This field is 1 if the file is younger than any of the jobs that need it. Young nodes are shown with a lighter shade of green in the GUI.
Z	Integer	NODES	The Z coordinate of a node (always 0).
ZIPPED	String	FILES	The string 'ZIP' if the file is zipped, or the empty string.

Selection Rules

Selection rules are used to create sets and to perform queries.

A selection rule consists of a list of predicates, separated by either spaces or logical operators. A non-quoted, non-escaped space between predicates is equivalent to an AND operator. Parentheses may be used to group logical operations or manipulate precedence. Each predicate typically consists of three parts:

1. The name of the field, which is required. The name of the field is case-insensitive and may contain extra underscores. For example, `ISJOB`, `IsJob` and `is_job` are all legal names for the field `ISJOB`.
2. An operator on the field. Refer to the list of operators.
3. A value. This part is interpreted as an integer for boolean and integer fields, or as a string for string fields. The value may contain spaces or other special characters if it is enclosed in double quotes (see [Examples of Selection Rules](#), below). Spaces in a value may also be entered if they are preceded by a backslash ("`\`") character. Operator and value can be omitted, in which case they default to `!=0` for numeric fields and `!=""` for string fields.

A special case is also supported where the name of a field can be passed by itself. This will query for objects that contain the field, and where the field's value is non-zero (for numeric fields) or non-empty (for string fields).

Supported logical operations are AND, OR and NOT. In the absence of parentheses, AND operations will always take precedence over OR operations at the same level. For example, the expression:

```
idint<5000 | idint>10000 & isjob
```

is evaluated as:

```
idint<5000 | (idint>10000 & isjob)
```

which may not be what was intended. To make sure the `isjob` predicate applies to the entire rule, use parentheses to group the predicates explicitly:


```
(idint<5000 | idint>10000) & isjob
```

When selecting multiple values from a single field, comma-separated lists of values are supported. For example to select all INVALID and FAILED jobs, the selection rule can be written as:

```
isjob & status==FAILED,INVALID
```

This rule is equivalent to:

```
isjob & (status==FAILED | status==INVALID)
```

 **Note:** Commas used in regular expressions, that is, with the `~`, `^` or `:` operators, will be interpreted as separators in a list of regular expressions. For example, the rule `isjob status~A,B` will match any job with "A" or "B" in its status field and does NOT attempt to match the string "A,B". If you wish to use a comma in a regular expression, enclose the expression in quotes. In this example you would use `isjob status~"A,B"`.

Selection rules that accept integer values will also accept timespecs; for example `"isjob autokill>10m"` would select all jobs with an autokill set to greater than 10 minutes.

Examples of Selection Rules


Selection Rule	Explanation
<code>isjob==1</code>	Select all jobs
<code>isjob</code>	Select all jobs (equivalent to <code>"isjob!=0"</code>)
<code>!isjob</code>	Select everything except jobs (equivalent to <code>"isjob!=0"</code>)
<code>not isjob</code>	Select everything except jobs (equivalent to <code>"isjob!=0"</code>)
<code>IS_FILE and db!=FILE</code>	Select all files which are not in the database <code>FILE</code>
<code>status==RUNNING</code>	Select all nodes whose status is <code>RUNNING</code>
<code>status==RUNNING,VALID</code>	Select all nodes whose status is <code>RUNNING</code> or <code>VALID</code>
<code>status!=RUNNING,VALID</code>	Select all nodes whose status is neither <code>RUNNING</code> nor <code>VALID</code>
<code>status!=RUNNING and status!=VALID</code>	Select all nodes whose status is neither <code>RUNNING</code> nor <code>VALID</code>

Selection Rule	Explanation
<code>isjob status==INVALID</code>	Select all invalid jobs
<code>isjob duration>300</code>	Select all jobs that have lasted more than 5 minutes (i.e. 300 seconds).
<code>isjob command~~aa</code>	Select all jobs with a command line containing the string <code>aa</code> ; the <code>~~</code> operator is used for case insensitive match.
<code>isjob age<600</code>	Select all jobs completed less than 10 minutes ago.
<code>isjob & age<600</code>	Select all jobs completed less than 10 minutes ago.
<code>isjob AND age<600</code>	Select all jobs completed less than 10 minutes ago.
<code>isfile name~ccc</code>	Select all files with name containing <code>ccc</code> .
<code>isjob inputs<2 status==INVALID</code>	Select all jobs that have fewer than 2 inputs and are invalid.
<code>isjob AND (inputs<2 OR status==INVALID)</code>	Select all jobs that have fewer than 2 inputs or are invalid.
<code>isjob (inputs<2 status==INVALID)</code>	Select all jobs that have fewer than 2 inputs or are invalid.
<code>isjob inputs<2 status==INVALID</code>	Select all jobs that have fewer than 2 inputs, and all nodes (including non-jobs) that are invalid. See notes on AND/OR precedence, above.
<code>isfile status==VALID age>=600 name~xxx</code>	Select all valid files older than 10 minutes whose name contains the string <code>xxx</code> .
<code>isfile status==VALID age>=10m name~xxx</code>	Select all valid files older than 10 minutes whose name contains the string <code>xxx</code> .
<code>isjob tool==gcc resources~diskio duration>10</code>	Select all jobs that use the tool <code>gcc</code> and require the resource <code>diskio</code> and take more than 10 seconds.
<code>isjob status~A,D</code>	Select all jobs that have either "A" or "D" in their Status fields; see notes on commas in regular expressions, above.

Selection Rule	Explanation
<code>isjob & status~"A,D"</code>	Select all jobs that have the string "A,D" in their Status fields.
<code>isjob & status~[A-Z]+A[A-Z]+D</code>	Select all jobs that match the regular expression "[A-Z]+A[A-Z]+D" in their Status fields (e.g. FAILED, INVALID).
<code>isjob status::inv*</code>	Select all jobs whose status fields start with "inv" (case-insensitive).
<code>isjob and status:inv*</code>	Select all jobs whose status fields start with "inv" (case-sensitive).
<code>isjob status!::inv*</code>	Select all jobs whose status fields do not start with "inv" (case-insensitive).
<code>isjob AND command!^"sleep 60"</code>	Select all jobs whose commands do not contain the string "sleep 60" (case-sensitive). Note that spaces are allowed in quoted values.
<code>isjob AND command!^sleep\ 60</code>	Select all jobs whose commands do not contain the string "sleep 60" (case-insensitive). Note that spaces are allowed if preceded by a backslash ("\").

Selection Rules and Incompatible Fields

A predicate based on an incompatible field is always true. Thus, the effect of the rule `isjob name~xxx` is to select all jobs in the trace, because the predicate `isjob` is true for jobs and false for files, while the predicate `name~xxx` is true for all jobs because the field "NAME" is incompatible for jobs.

 **Note:** The flag for skip is actually named `autoflow`.

For example:

```
# Look for skipped jobs
isjob ISAUTOFLOW==1
```

Selection Rules Operators

Operators that can be applied to fields.

Operator	Meaning	Numeric	Boolean	String	Predicate	Comment
=	Equal	*	*	*		Equivalent to ==
==	Equal	*	*	*		Equivalent to =
!=	Not equal	*	*	*		
>	Greater	*				
>=	Greater or equal	*				
<	Less	*				
<=	Less or equal	*				
~	Matches			*		Uses regular expressions
!~	Does not match			*		Uses regular expressions
~~	Matches (case-insensitive)			*		Uses regular expressions
!~~	Does not match (case-insensitive)			*		Uses regular expressions
^	Substring			*		Faster than the match(~) and glob(:) operators.
!^	Not Substring			*		Faster than the match(~) and glob(:) operators.

Operator	Meaning	Numeric	Boolean	String	Predicate	Comment
^^	Substring (case-insensitive)			*		Faster than the match(~) and glob(:) operators.
!^^	Not Substring (case-insensitive)			*		Faster than the match(~) and glob(:) operators.
:	Matches glob expression			*		Faster than regular expressions
!:	Does not match glob expression			*		Faster than regular expressions
::	Matches glob expression (case-insensitive)			*		Faster than regular expressions
!::	Does not match glob expression (case-insensitive)			*		Faster than regular expressions
&	Logical AND				*	Takes precedence over . Equivalent to AND
AND	Logical AND				*	Takes precedence over OR. Case-insensitive; equivalent to &
	Logical OR				*	Lower precedence than &.

Operator	Meaning	Numeric	Boolean	String	Predicate	Comment
						Equivalent to OR
OR	Logical OR				*	Lower precedence than AND. Case-insensitive; equivalent to
!	Logical NOT				*	
NOT	Logical NOT				*	Case-insensitive
()	Parentheses				*	Used for grouping logical expressions

Formatting Strings

A formatting string is a string that contains field references; it is used to list the elements of a set.

A *field reference* consists of the name of a [field](#), which can include an optional size specification (integer value).

- The syntax for a field reference: @field@ or @field:size@.
- The size can be used to format tables and reports.

Field Reference

@STATUS@ is a reference for the Status field. The name of the field in formatting strings is case insensitive and can contain underscores. The commands in the following example yield the same result:

```
% vovset show -O "@id@ @status@" System:jobs
% vovset show -O "@ID@ @STATUS@" System:jobs
% vovset show -O "@Id@ @S_t_A_t_U_s_@" System:jobs
```

Size Specification

The size specification consists of a colon and an integer and is used to format tables and reports.

- If the field value is shorter than the specified size, the value is padded to the right with spaces in order to reach the desired size.

- If the size is negative, the width of the formatted field is the absolute value of the size and the field is right-justified.
- If the size is positive, the field is left-justified.
- If the size is zero, the field is not truncated.
- If the field value is shorter than the specified size, the value is padded to the right with spaces in order to reach the desired size.
- If the size is negative, the width of the formatted field is the absolute value of the size and the field is right-justified.
- If the size is positive, the field is left-justified.
- If the size is zero, the field is not truncated.
- If the absolute value of the size cannot exceed 1000.
- If the absolute value of the size cannot exceed 1000.
- Any character between the size specification and the second @ sign is silently dropped.

Each field reference is replaced by the corresponding value of the field for each node. If the field is incompatible for a node (for example, if a job node does not have a NAME field) the reference is replaced with the empty string.

For example:

```
% vovset show -O "id=@ID@ (@STATUS@)" System:jobs
id=001234567 (VALID)
% vovset show -O "id=@ID@ (@STATUS:12@)" System:jobs
id=001234567 (VALID )
% vovset show -O "id=@ID@ (@STATUS:-12@)" System:jobs
id=001234567 ( VALID)
% vovset show -O "id=@ID@ (@STATUS:-12extra_chars_that_are_dropped@)" System:jobs
id=001234567 ( VALID)
```

Failure Codes

When a job fails, some explanation for the failure is stored in a field called "FAILCODE" of type integer.

This field consists of a bit mask, where the bits are explained by the table below, which is generated by the command `vovshow -failcodes`.

Hex	Octal	Dec	Explanation
0x00000001	00000001	1	Preempted by owner
0x00000002	00000002	2	Preempted by admin
0x00000004	00000004	4	Preempted by system
0x00000008	00000010	8	Withdrawn
0x00000010	00000020	16	Killed by owner
0x00000020	00000040	32	Killed by administrator
0x00000040	00000100	64	Killed by system
0x00000080	00000200	128	Killed by autokill
0x00000100	00000400	256	Bad exit status
0x00000200	00001000	512	Got a signal
0x00000400	00002000	1024	Cannot change directory
0x00000800	00004000	2048	Bad environment
0x00001000	00010000	4096	No executable

0x00002000	00020000	8192	No outputs
0x00004000	00040000	16384	Input dependency issue
0x00008000	00100000	32768	Output conflict
0x00010000	00200000	65536	Cannot switch user id
0x00020000	00400000	131072	Cannot switch group id
0x00040000	01000000	262144	Cannot start
0x00080000	02000000	524288	Migrated
0x00100000	04000000	1048576	Bad timestamp of outputs
0x00200000	10000000	2097152	Pre-condition Failed
0x00400000	20000000	4194304	Post-condition Failed
0x00800000	40000000	8388608	Bad X11 Display
0x01000000	100000000	16777216	Too many ORs in resources
0x02000000	200000000	33554432	Bad resources
0x04000000	400000000	67108864	Failed to get Unix groups
0x08000000	1000000000	134217728	Killed by authorized user

This field is mostly used:

- for reporting using the SQL database
- to distinguish a job that has been withdrawn by preemption from other type of failures. These jobs have the flag 0x8 set (withdrawn), and the jobs with that flag have the status "WITHDRAWN".

VOV Metrics

The Altair Accelerator allows users to collect arbitrary *metrics*, where a metric is defined as a "named time series" of values, where each value is either a floating point number or a string.

There are three families of metrics:

- Server metrics are collected automatically by the vovserver, and capture information about the scheduler, like the number of queued jobs or the jobs-per-hour statistics.
- Design metrics are controlled by the user.
- FairShare metrics.

By default, all three families of metrics are enabled. To enable a subset, set the server configuration parameter `metrics.enable` to a decimal mask constructed from this bit components:

- 1 = `enable_server_metrics`
- 2 = `enable_design_metrics`
- 4 = `enable_fairshare_metrics`

To activate all metrics, use the value 7.

Metrics Storage

The metrics are primarily stored in the vovserver memory. You can choose how much RAM is to be used for the metrics using the parameter `metrics.maxmem`, which is a value in MB.

When the total RAM used by the metrics exceeds this value, metrics will be reduced by a combination of methods, chosen automatically by vovserver:

- Cropping of old data
- Reduction of accuracy of old data
- Cropping metrics that are not recently accessed

The metrics are saved from time to time to files in the server working directory, precisely under the directory `PROJECTNAME.swd/data/metrics`. One or more files are used for each metric. The name of the files is the name of the metric itself, possibly extended with a timestamp.

Metrics Naming

The name of a metric determines the name of the file used to store the metric. Therefore, the name of a metric may include any character that is acceptable in a file name.

Simple metrics may have names like "myMetric" "xxx". The server metrics have names like "scheduler/runningJobs", "scheduler/slots/inuse", or "scheduler/jobsPerHour".

The slashes in a name separate the components of the name. If a component has the form PROPERTY=VALUE, then the PROPERTY part identifies a property of the metric which can be queried. For example, a metric with name

```
site=US/user=john/res=abc/inuse
```

can be considered as a metric with 4 components and 3 properties, namely "site" "user" and "res".

Built-in Server Metrics

```
% vovmetric list
1 031776922 SERVER scheduler/queuedJobs integer
2 031776923 SERVER scheduler/runningJobs integer
3 031776924 SERVER scheduler/slots/available integer
4 031776925 SERVER scheduler/slots/capacity integer
5 031776926 SERVER scheduler/slots/inuse integer
6 031776927 SERVER scheduler/ram/available MB
7 031776928 SERVER scheduler/ram/capacity MB
8 031776929 SERVER scheduler/ram/inuse MB
9 031776930 SERVER scheduler/jobsPerHour
jobsPerHour
10 031776931 SERVER scheduler/time ms
11 031776932 SERVER trace/files integer
12 031776933 SERVER trace/jobs integer
13 031776934 SERVER trace/sets integer
14 031776935 SERVER process/size MB
15 031776936 SERVER licmon/users/observed integer
16 031776937 SERVER licmon/users/licensed integer
```

VOV Metrics Command Line Interface

The simplest way to manage metrics is with the utility `vovmetric`.

`vovmetric`: Usage Message

DESCRIPTION:

Manipulate VOV metrics. Metrics are organized by family. The SERVER family is available out of the box. The DESIGN family must be populated by the user. FAMILY can be anything as long as it is upper case.

USAGE:

```
% vovmetric ACTION [OPTIONS]
```

```

OPTIONS:
  -h                -- This help.
  -v                -- Increase verbosity.
  -family           -- Requires -family <FAMILY> to be passed when not
                    using the default. <FAMILY> must be upper case.
                    (default DESIGN)

ACTIONS:
  enable            -- Enable the metric system.
  disable           -- Disable the metric system (all metrics are lost).
  usedb            -- Set to:
                    -0 = To use metrics from in-memory metrics database
                    -1 = To use metrics from time-series database
  add              -- Add point to a metric, creating metric
                    if necessary.
  get              -- Get info about a metric.
  list             -- List all metrics; same as 'show'.
  show            -- Same as 'list'.
  delete          -- Delete specified metric.
  save           -- Save specified metric to disk. Metric data files
                    are stored per family in SWD/data/metrics.
  status         -- Show metric subsystem status:
                    -1 = enabled, but unlicensed
                    0 = disabled
                    N = enabled, masked against metric family

EXAMPLES:
% vovmetric add -name JobsToGo -value 33
% vovmetric delete -name JobsToGo
% vovmetric list -rx Jobs
% vovmetric get -name JobsToGo

```

Using vovselect

You can also query the metrics using vovselect.

```

% vovselect id,name from metrics
...this returns id and name of all metrics
% vovselect id,name from metrics where name~/ram/
id      name
000113670 scheduler/ram/available
000113671 scheduler/ram/capacity
000113672 scheduler/ram/inuse

```

If you want to access a property of a wave (reminder: a property is computed from the metric name, which may consist of components of the form PROPERTY=VALUE), use the field 'PROP.<propertyName>' as in this example:

```

% vovselect id,name,prop.user from metrics where prop.site==US
000150881 site=US/user=john/res=abc/m john
000150882 site=IN/user=john/res=abc/m john

```

Run `vovselect fieldname,fieldtype from <object>` to see the list of fields for that object. Multiple fields may be requested by separating them with a comma. Some fields represent a data collection that can be broken down using a format of `FIELD.X`, such as:

```
KEY.<KEYNAME>      (metric objects)
PARAM.<PARAMNAME>  (server object)
PROP.<PROPNAME>    (all objects)
RESOURCES.<RESNAME> (tasker objects)
```

To find the fields that are accessible for metrics, you can use this command:

Usage: `'vovselect fieldname from metrics'`

```
AGE
AGEPP
FAMILY
FIRSTTS
ID
IDINT
KEY
LASTACCESS
LASTDATA
LASTLABEL
LASTTS
LASTUPDATE
MEMORY
NAME
POINTS
PROP
PROPDUMP
PROPERTIES
ROWCOUNT
TAIL
UNITS
```

Vtk Interface

The low-level interface for metrics is with the `vtk_metric` family of procedures.

```
vtk_metric_add
vtk_metric_get
vtk_metric_getwave
vtk_metric_find
vtk_metric_forget
```

Plotting

The `vovconsole` has some ability to plot metrics using the `VovPlotMetrics` procedure.

Miscellaneous

Inner Loop Timers

The vovserver tracks how the time is spent in the inner loop. The statistics are accumulated over multiple loops and are rotated every 10 seconds, so what you normally see in the stats is the overall time spent in the past 10 to 20 seconds.

If you are ADMIN, you can check the timers with `vovshow`:

```
% vovshow -innerlooptimers
          PHASE  TGT    ACTUAL  TIME (ms)    MAX (ms)
INNER_LOOP_TIMER  idle    0    22.91   2931.476    0.032
INNER_LOOP_TIMER  poll    0     2.84    362.885    0.001
INNER_LOOP_TIMER  misc    0     0.09    11.580    0.000
INNER_LOOP_TIMER  tasks   20     1.42    181.897    0.010
INNER_LOOP_TIMER  sched   20     0.38     49.108    0.001
INNER_LOOP_TIMER  work    0    48.04   6146.102    0.222
INNER_LOOP_TIMER  http    0     0.01     0.961    0.000
INNER_LOOP_TIMER  fairshare 0     0.00     0.601    0.000
INNER_LOOP_TIMER  preempt 0     0.00     0.004    0.000
INNER_LOOP_TIMER  notify   35    24.31   3110.120    0.168
INNER_LOOP_TIMER  total    0   100.00  12794.734    0.000
```

Here is a brief explanation of the timers in the inner loop:

Timer Name	Description
idle	Time spent waiting for clients. In the code, this is the time spent in <code>poll()</code> , <code>epoll_wait()</code> , or <code>select()</code> . If the vovserver spends a lot of time in "wait" then there should be no performance issue at all.
poll	Time spent preparing the poll-array or choosing the next client to service.
misc	Time spent miscellaneous activities.
tasks	Time spent in server-initiated tasks, such as checking smart sets and removing expired sets. This activity is controlled by the parameter <code>tasksMaxEffort</code> .
sched	Time spent dispatching jobs from buckets to taskers. This activity is controlled by the parameter <code>schedMaxEffort</code> .
notify	Time spent sending events to clients and saving events to the journal and to the crash recovery file. This activity is controlled by the parameter <code>notifyMaxEffort</code> .

Timer Name	Description
fairshare	Time spent updating FairShare statistics
work	Time spent servicing clients (except HTTP clients)
http	Time spent servicing HTTP clients
total	Total time spent in inner loop

Sequence of Firing of a Job on a Tasker

This is the sequence to fire a job using `vovtaskerroot` running on UNIX.

1. The tasker is running as root

- The tasker receives from vovserver a request to start a job. The request contains the information about the job and about all the properties attached to the job, including PRECMD and POSTCMD.
- If the flag `-E` of vovtasker is used, then change command line of command to execute to `"vovfire $JOBID -l SOME_LABEL > /tmp/vovfire.$JOBID.log`.
- If a PTY is requested, create the PTY and connect to the process on the submission host.
- Compute affinity mask if requested (NUMA control).
- Make sure all groups for the executing user have been cached.
- `fork()` a subtasker. The parent process goes back to the main loop. The child, that is, the subtasker, will be used to shepherd the job.
- The subtasker sets its own affinity mask (if required).
- The subtasker creates the PTY and connects it to the submission process.
- If `VOV_DEBUG_TASKER` is set, the subtasker sleeps 10 seconds (to allow connection of a debugger)
- The subtasker tries 3 times to switch user identity (`uid` and `gid`). In each attempt,
 - it switches `gid` with `setgid()`
 - it switches `uid` with `setuid()`
 - it rebuilds the environment for the user (`HOME`, `USER`, `LOGNAME`, `SHELL`)

If the switch fails, the subtasker waits 10 seconds before the next attempt.

- The subtasker sets signals `XCPU` `XFSZ` `PIPE` `USR1` `USR2` to their default behavior.
- The subtasker calls `nice(8 - execPriority)`, based on the value of the execution priority for the job.
- If the switch of user identity fails, the system calls the diagnostics script `vov_diagnostics_setuid` (not as root, but as the owner of Accelerator).
- Now the subtasker is running as the user that owns the job.**
- The subtasker tries to change directory `chdir()`. If it fails the first time, it tries a few more times based on `VOV_RETRY_CHDIR` and `VOV_RETRY_CHDIR_SLEEP`.

17. If the directory cannot be changed, the subtasker calls `vov_diagnostics_chdir` with arguments ID and DIR (the directory that could not be accessed)
18. The subtasker tries to switch environment
19. The subtasker executes the `.pre.` scripts of the environment. (obsolete, but still there)
20. The subtasker executes the `precmd` script with a `system()` call. If the precommand exits with a status that is not 0 (zero), the job is done and failed.
21. The subtasker executes the job and waits for it to finish.
22. The subtasker executes the `postcmd` script with another `system()` call. The exit status of the `postcmd` is used as exit status of the job.

Error Message: "Too many notify clients for user ..."

The clients (GUI, monitors, and `-wl` or `nc wait`) that are listening to the Accelerator vovserver's event stream are called *"notify clients"* and they take two file descriptors on the vovserver.

To prevent accidental or malicious denial-of-service attacks, the vovserver limits the number of regular and notify clients. The default for notify clients is 40, and it is controlled by a `config(maxNotifyClients)` in `vnc.swd/policy.tcl`.

```
# In policy.tcl
set config(maxNotifyClients) 100; ## Increase from default 40.
```

As with all configuration parameters, it is also possible to change this parameter on the fly with:

```
# Example of changing a parameter from the command line.
% vovsh -x 'vtk_server_config maxNotifyClients 100'
```

Reaching the limit of notify clients usually indicates that jobs are being submitted incorrectly. For example, you may have a user that submits a blocking job (that is, one that uses the option `-wl` or the option `-w`) and then puts that blocking job in the background. The waiting for the job to complete is done by tapping the event stream and therefore uses a NOTIFY client. If this is done in a script with tens or hundreds of jobs, then each one uses a NOTIFY client and the user may hit the limit.

Ask the user how he/she is submitting jobs and review the submission line. If the submission uses options `-wl`, `-w` or `-Ir`, evaluate whether that is really necessary. There may be more efficient ways to submit many jobs and then to wait for them to complete, for example using sets (see `nc run -set SETNAME ...`) and then waiting for the whole set with `nc wait -set SETNAME`.

Examples

Look at this submission:

```
% nc run -Ir ls -R &
```

The option `-Ir` implies waiting for the job to complete, so that we can show all I/O on the submission side, which normally requires tapping the event stream, while the final ampersand `&` implies "do not

wait" (ls -R is just an example of a command). If this is done for many jobs, you may exceed the maximum number of NOTIFY clients.

The recommended alternative would be to submit all jobs as non-interactive jobs and then wait for them:

```
% nc run -set MyUniqueSetName ls -R
% nc run -set MyUniqueSetName ls -R
...
% nc wait -set MyUniqueSetName
```

Debugging Tip

A useful method to debug the submission of jobs is to use the VOV_DEBUG_ORIGARGS environment variable. This will put a property named ORIGARGS on each job that will contain the submit command. Do not use this function unless it is needed, however, since it will cause the server to consume some extra amount of RAM with higher workloads.

Error message: "Cannot accept more connections"

This message, which can be found in the vovserver log, means that the table of connection is filling up and the vovserver refuses to accept more. This is connected to the number of file descriptors available to vovserver. This is typically around 1,000 for small compute farms or 65,000 and even more for large compute farms.

The remedy could be:

- Find out if there are abusers of the system. You can do that with:

```
% vovshow -users
% vovshow -clients
```

If an abuser is identified, perhaps he or she can be educated on the socially responsible use of the computing resources.

- Increase the number of file descriptors available to the vovserver. Check the *Server Capacity* page in the FlowTracer Administration Guide for more details. This will require a restart of vovserver.

vtk_server_config

This section explains the usage of vtk_server_config. Only an ADMIN is allowed to use this API.

```
% vovsh -x 'vtk_server_config KEYWORD VALUE'
```

All configuration parameters can be controlled via this interface. In addition, the following keywords are understood.

KEYWORD (case-insensitive)	VALUE	Description
allowcoredump	boolean	If the vovserver encounters a serious error, it may dump its core if the flag is set.
autoshutdown	time_spec	Control the auto-shut-down time for the vovserver. The vovserver shuts down when the specified time elapses after the last executed job. If the value is 0, auto-shut-down is disabled.
chdir	directory_path	Change working directory of the vovserver. This can be useful if the OS loses the notion of where the vovserver is running (seen on some old systems).
checkbarrier	ignored	Run a sanity check for barriers, including the barrier-invalid flags. Only useful for FlowTracer.
checklogchecklogs	size	Check the size of the vovserver log and rotate it if it is larger than the specified size.
clearsymlinks	ignored	Force clearing of the caches for symlinks. See also printsymlinks.
clientfdoffset	integer	DEBUG: do not use (default is 40).
closeclientbyhost	hostname	Close all clients on the specified host. You can see the connected clients with <code>vovshow -clients</code> . This can be used, for example, when the OS is refusing to shut-down a socket even when a machine is down. See also <code>closeclientsbyindex</code> .
closeclientbyindex	integer	Close the specified client. The value is the index of the client. You can see the connected clients with <code>vovshow -clients</code> . This can be used, for example, when the OS is refusing to

KEYWORD (case-insensitive)	VALUE	Description
		shut-down a socket even when a machine is down. See also <code>closeclientsbyhost</code>
<code>dumphalfattachments</code>	<code>ignore</code>	DEBUG: Create a file called <code>ha.dump</code> to show where the half-attachment memory is being spent.
<code>enablepam</code>	<code>boolean</code>	
<code>enableldap</code>	<code>boolean</code>	
<code>enterpriselicensecoreslots</code>	<code>integer</code> <code>"off"</code> <code>"auto"</code> <code>"full"</code>	Control the use of the enterprise license. See also <code>"refreshlicense"</code> below for more interaction with licensing.
<code>flushlogs</code>	<code>ignore</code>	Force flushing of all journals and logs. (normally all logs are flushed once a second).
<code>graceperiod</code>	<code>time_spec</code>	DEBUG ONLY: set the grace period for when the license expires. The default value is 5d, the range is between 1m and 5d.
<code>httpsecure</code>	<code>boolean</code>	If the flag is 1 (which is the default), one must authenticate to use the browser interface. Otherwise everyone can login and browse using <code>READ_ONLY</code> credentials. See also environment variable <code>VOV_HTTP_UNSECURE</code> .
<code>idmgmt</code>	<code>vovid</code>	Test the insertion of an object with specified VovId. If the VovId is large, this will likely trigger the recycling of VovIds. Used mostly for testing.
<code>idrange</code>	<code>lowId-highId</code>	Specify the range of VovId recycling.
<code>liverecorder</code>	<code>"on"</code> or <code>"off"</code> or PID of vovserver	Controls the live recording capability provided by Undo Software . Turning live recorder

KEYWORD (case-insensitive)	VALUE	Description
		on is quick, turning it off requires a relatively long time (about 30s if you have 200k jobs in the system)
maxclients	integer	DEBUG: used to test the behavior of system when a small number of clients are allowed. Reduces the total number of usable clients. Internal levels are enforced.
maxlevel	integer	Used in FlowTracer. Set the maximum level for the flow. The default value is 4095. The range is 3 to 4095.
maxjobarray	integer	Used in Accelerator. Set the maximum size of a job array. The default is 10,000. The range is 10..100,000.
maxnormalclients	integer	
maxnotifyclients	integer	Maximum number of notify clients per user. This is a protection against denial-of-service attacks by a single user. The 'notify' clients are those that tap into the event stream. They are used for example in waiting for jobs to finish and to update the GUI.
maxresmap	integer	
maxbuffersize	integer	
minhw	string	Set the value of the minimum hardware resources that every job needs to request. The default value is "RAM/20 CORES/1 SLOTS/1 PERCENT/1".

KEYWORD (case-insensitive)	VALUE	Description
netinfo	boolean	Control whether the system is supposed to collect network information from all its taskers.
preemptionfairshare	"on" "off" time_spec	Enable or disable the computation to support FairShare of preemption.
printcodes	ignored	Force printing usage of RPC codes (debugging only). Output goes to the vovserver log.
printsymmlinks	ignored	Force printing of content of symlink cache (debugging only). Output is in vovserver log. See also clearsymmlinks.
product	lm lms nc mq la ft rm	Set the product of vovserver.
readonlyport	integer	Control the read-only port for vovserver. If the integer is 0, the read-only port is closed. If in the range 1000...32000, the specified port is opened, if possible. If the port cannot be opened, then the port is closed again.
reset_debug_flag	flag-name	DEBUG: turn OFF the named debug flag. The list of flags is not published. See also set_debug_flag.
reopenlogs	ignore	Close and reopen all journals and logs. See also checklogs, flushlogs, and rotateserverlog.
rotateserverlog	boolean	Rotation of the server logs, but only if the boolean is true.
scheduler	command	Request a scheduler cycle. If the command is the word "suspend", then the scheduler is suspended, meaning no jobs are dispatched. If the command is "request", then the scheduler is called once, while remaining suspended

KEYWORD (case-insensitive)	VALUE	Description
		or active. If the command is "unsuspend", then the scheduler is reactivated. If the command is anything else, the scheduler is reactivated and called once.
set_debug_flag	flag-name	DEBUG: turn ON the named debug flag. The list of flags is not published. See also reset_debug_flag.
taskerheartbeat	num_seconds	Set the expected heart beat of the taskers. If a tasker is not reporting an update after 3*taskerheartbeat, it is considered SICK. The range for this is between 10 and 86400(1day).
trace_debug_flag	bit-mask	DEBUG: Set some debug flags in VovTrace. Default value is 0. See also VOV_DEBUG_FLAGS.
type	string	Set the project type.
usrtmp	directory	Specify the directory to be used for temporary files. Default is /usr/tmp or /tmp.

Expected Duration

The expected duration of a job is the *duration of the most recent successful execution* of the job. The job's expected duration may also be specified by the user.

In FlowTracer this is done in the `Flow.tcl` script using the procedure [X](#). Accelerator users can specify the expected duration of a job with the option `-xdur <TimeSpecification>` of `nc run`.

If the user has not specified an expected duration and the job has not yet completed successfully, its expected duration is unknown, represented by -1.

The current expected duration can be accessed in formatting strings using the fields [XDUR](#) and [XDURPP](#).

The last expected duration specified by the user, if any, can be accessed in formatting strings using the fields [USERXDUR](#) and [USERXDURPP](#). Note that these values are intended for reference purposes and may differ from `XDUR` and `XDURPP`.

Environment Variables

Variable Name	Used By	Description
<i>CLEARCASE_ROOT</i>		
<i>DISPLAY</i>	vovsh GUI nc run -Ix	Name of the X display. It is required to use the graphical user interface.
<i>HERO_EMUL</i>	HERO hero	The name of a specific emulator in Hero. Available at runtime
<i>HERO_HOST</i>	HERO HERO_PORT	Together with HERO_PORT, used by some Hero utilities to locate the HERO wrapper for one emulator. Each emulator has a wrapper and the wrapper can be found with HERO_PORT and HERO_HOST. This variable should not be set by the user.
<i>HERO_PORT</i>	HERO HERO_HOST	Together with HERO_HOST, it is used by some Hero utilities to locate the HERO wrapper for one emulator. Each emulator has a wrapper and the wrapper can be found with HERO_PORT and HERO_HOST. This variable should not be set by the user.
<i>HERO_VENDOR</i>	HERO hero	One of the supported Hero vendors (currently, one of zebu palladium veloce).
<i>KRB5CCNAME</i>	nc kerberos	The name of the Kerberos Credential Cache file.
<i>LD_LIBRARY_PATH</i>	vrt	
<i>LD_LIBRARY_PATH_64</i>	vrt	
<i>LD_PRELOAD</i>	vrt	Used on Linux to preload <i>vovrti.so</i> , the interception library.
<i>LM_PROJECT</i>	nc	This variable is used by FlexNet Publisher to specify a project for a checkout. It is also used by <i>nc run</i> to define the default value of the <i>jobproj</i> field of a job. See also VOV_JOBPROJ and RLM_PROJECT.
<i>LM_VAR_NAME</i>	vovgetflexlmdaemons vwrap	Specify the name of the environment variables to be set on the basis of the grabbed resources. This variable is used by the wrapper <i>vw</i> and by the

Variable Name	Used By	Description
	vw	script <code>vovgetflexlmdaemons.tcl</code> . The variable <code>VOV_LM_VARNAMES</code> is preferable to this.
<code>LOGNAME</code>		Normally defined in the standard environment (POSIX).
<code>NC_AUDIT_FILE</code>	nc	Specify a file in which to log NC CLI commands for auditing. The file must be writable by the user issuing the <code>nc</code> command. The variable may contain a single reference to another, which will be expanded to compute the file pathname. <div># This could go in the .cshrc file. Note single-quotes here. setenv NC_AUDIT_FILE '\$HOME/.vncaudit.txt'</div>
<code>NC_CONFIG_DIR</code>	nc/wx utilities ncmgr/wxmgr LSF emulation utilities	Specify the directory that contains the NC/WX queue configuration file. This location must be a shared network location that all clients can read. The default location is <code>\$VOVDIR/local/vncConfig</code> . Because the configuration is created by <code>ncmgr/wxmgr</code> , the variable must be set before running those utilities to create a queue. Afterward, the variable must be set in the admin-defined, shell-specific <code>vovsetup.csh sh tcl</code> script (Linux) in <code>\$VOVDIR/local</code> , or <code>vovinit.bat</code> (Windows) in <code>\$VOVDIR/local</code> .
<code>NC_DEFAULT_JOBCLASS</code>	nc run nc jobclass	This is a useful variable to speedup job submission with <code>nc run</code> . If this variable is set, use its value to find out the default jobclass. This is should be the same as the property <code>NC_DEFAULT_JOBCLASS</code> attached to Object 1.
<code>NC_FIFO</code>	nc run	Use this variable if you are running a lot (thousands) of <code>nc</code> commands from the same shell. This variable points to a file that is used in a <code>mkfifo</code> command to create a fifo. When <code>nc</code> is run, it checks that fifo. If it does not exist, it creates it, and then waits for commands to arrive on that fifo. If it does exist, it sends the arguments of <code>nc run</code> to the fifo. The benefit is that we bypass the initialization of the Tcl interpreter. <div># Example: # This could go in the .cshrc file setenv NC_FIFO /tmp/vovncfifo\$\$</div>
<code>NC_LIST_FORMAT</code>	nc	Specify the default format for <code>nc list</code> .

Variable Name	Used By	Description
<code>NC_LOGDIR</code>	<code>nc run</code> <code>bsub</code>	<p>Specify the directory to be used to store the logs. This is used also by the LSF emulation script <code>bsub</code>.</p> <pre># This could go in the .cshrc file setenv NC_LOGDIR \$USER/.myvnc_logs</pre> <p>Old names also supported: <code>VNC_LOGDIR</code>, <code>FTNC_LOGDIR</code>.</p>
<code>NC_LOGDIR_PERMS</code>	<code>nc run</code> <code>bsub</code>	<p>Specify the permissions for the directory to be used to store user logs. Must be specified in UNIX permissions format. Default value is 0775. This is used also by the LSF emulation script <code>bsub</code>.</p> <pre># This could go in the .cshrc file setenv NC_LOGDIR_PERMS 0777</pre>
<code>NC_NOLOG</code>	<code>nc run</code> <code>bsub</code>	Internal variable used only to disambiguate the environment for interactive jobs. Do not set this variable.
<code>NC_OLDQUEUE</code>	<code>nc</code> <code>ncmgr</code>	Used to migrate from an old queue to the new queue.
<code>NC_OLDVERSION</code>	<code>nc</code> <code>ncmgr</code>	Used to migrate from an old queue to the new queue.
<code>NC_QUEUE</code>	<code>nc</code> <code>ncmgr</code> <code>wx</code> <code>wxmgr</code>	Specify the name of the queue used by the <code>nc</code> and <code>wx</code> commands. This can be specified as a simple name (e.g. XXXX), in which case the system sources the file in <code>\$VOVDIR/local/vncConfig/XXXX.tcl</code> , or it could be the full path to the <code>setup.tcl</code> file in the <code>.swd</code> . If not specified, the value used is "vnc".
<code>NC_RUN_ARGS</code>	<code>nc</code>	Specify the default arguments for <code>nc run</code> . The arguments in this variable are prepended to the arguments passed on the command line. Use <code>NC_RUN_ARGS_AFTER</code> if you want to append arguments to the command line.
<code>NC_RUN_ARGS_AFTER</code>	<code>nc</code>	Specify some default arguments for <code>nc run</code> . The arguments in this variable are appended to the arguments passed on the command line. Use <code>NC_RUN_ARGS</code> if you want to prepend arguments to the command line.

Variable Name	Used By	Description
<code>NC_RUN_BLOCKING_STDOUT</code>	<code>nc</code>	<p>Experimental. Controls whether the stdout/stdin descriptors in an interactive job are to be set to non-blocking or not. Typical usage in situations like:</p> <pre>% env NC_RUN_BLOCKING_STDOUT=0 nc run -l my_command tee foo.log</pre>
<code>NC_SNAPSHOTDIR</code>	<code>nc</code> <code>ncmgr</code>	<p>Specify the directory to be used to store the environment snapshots. This variable can be the path to a directory or a symbolic value such as</p> <ul style="list-style-type: none"> "normal", which is the same as using <code>\$NC_LOGDIR/snapshots/\$USER/\$VOVARCH</code> "fullpath", which is the same as "normal" except a full path is used "serverdir", which uses <code>\$PROJECT.swd/snapshots/\$USER/\$VOVARCH</code> "homedir", which uses <code>~/.ncsnapshots/\$VOVARCH</code> <p>The default directory for the snapshot is <code>./vnc_logs/snapshots/\$USER/\$VOVARCH</code></p>
<code>NC_SNAPSHOTDIR_PERMS</code>	<code>nc run</code> <code>bsub</code>	<p>Specify the permissions for the directory to be used to store the environment snapshots. Must be specified in UNIX permissions format. Default value is 0775. This is used also by the LSF emulation script <code>bsub</code>.</p> <pre># This could go in the .cshrc file setenv NC_SNAPSHOTDIR_PERMS 0777</pre>
<code>NC_SNAPSHOTFILE_PERMS</code>	<code>nc run</code> <code>bsub</code>	<p>Specify the permissions for the environments snapshot files. Must be specified in UNIX permissions format. Default value is 0664. This is used also by the LSF emulation script <code>bsub</code>.</p> <pre># This could go in the .cshrc file setenv NC_SNAPSHOTFILE_PERMS 0644</pre>
<code>NC_STOP_SIGNALS</code>	<code>nc stop</code> <code>bkill</code>	<p>Specify the signals sent by <code>nc stop</code>. This is a comma-separated list of signals to be sent to the processes in the job. The signals include TERM INT KILL HUP USR1 USR2 CONT TSTP.</p> <p>In addition, EXT is supported in the format <code>EXT:SIGNAL:INCLUDERX:EXCLUDERX:SKIPTOP</code> (notice that we use the colon ':' instead of ',' commas).</p>

Variable Name	Used By	Description
		<ul style="list-style-type: none"> SIGNAL is required INCLUDEDERX is optional and is a regexp that matches names of processes that will be sent the signal EXCLUDERX is optional and is a regexp that matches names of processes that will NOT be sent the signal SKIPTOP is optional (default 1) and is a boolean to control whether the top process (normally <code>vw</code>) is skipped or not (i.e. by default we do not send the signal to <code>vw</code>) <p>Examples:</p> <pre>% setenv NC_STOP_SIGNALS TERM,INT,KILL % setenv NC_STOP_SIGNALS EXT:TSTP:vwish % setenv NC_STOP_SIGNALS CONT,CONT,EXT:TSTP:vwish,CONT</pre> <p>In the case of <code>nc stop</code>, instead of this environment variable, you can use a property on a job with the name <code>NC_STOP_SIGNALS</code> and with the same syntax. The signals are sent to the job with a delay determined by <code>VOV_STOP_SIGNAL_DELAY</code> or <code>NC_STOP_SIG_DELAY</code>, and is 3 second by default. If both <code>NC_STOP_SIGNALS</code> and <code>VOV_STOP_SIGNALS</code> are present in the environment, the value of <code>VOV_STOP_SIGNALS</code> will be used. If both <code>NC_STOP_SIG_DELAY</code> and <code>VOV_STOP_SIGNAL_DELAY</code> are present in the environment, the value of <code>VOV_STOP_SIGNAL_DELAY</code> will be used.</p>
<code>NC_STOP_SIG_DELAY</code>	<code>nc stop</code> <code>bkill</code>	Specify the delay between signals sent by <code>nc stop</code> . In seconds. The default value is 3 seconds. If both <code>NC_STOP_SIG_DELAY</code> and <code>VOV_STOP_SIGNAL_DELAY</code> are present in the environment, the value of <code>VOV_STOP_SIGNAL_DELAY</code> will be used.
<code>NC_SUSPEND_SIGNALS</code>	<code>vtk_job_control</code>	Specify the signals sent by <code>vtk_job_control</code> when action is SUSPEND This is a comma-separated list of signals to be sent to the processes in the job. The signals include TERM INT KILL HUP USR1 USR2 CONT TSTP. In addition, EXT is supported as a signal in the format <code>EXT:SIGNAL:INCLUDEDERX:EXCLUDERX:SKIPTOP</code> (notice that we use the colon ':' instead of ',' commas).

Variable Name	Used By	Description
		<ul style="list-style-type: none"> SIGNAL is required INCLUDERX is optional and is a regexp that matches names of processes that will be sent the signal EXCLUDERX is optional and is a regexp that matches names of processes that will NOT be sent the signal SKIPTOP is optional (default 1) and is a boolean to control whether the top process (normally <code>vw</code>) is skipped or not (i.e. by default we do not send the signal to <code>vw</code>) <p>Examples:</p> <pre>% setenv NC_SUSPEND_SIGNALS STOP % setenv NC_SUSPEND_SIGNALS EXT:STOP:::1</pre> <p>A property on a job with the name <code>SUSPEND_SIGNALS</code> can be used with the same syntax. If both <code>NC_SUSPEND_SIGNALS</code> and <code>VOV_SUSPEND_SIGNALS</code> are present in the environment, the value of <code>VOV_SUSPEND_SIGNALS</code> will be used. The signals are sent to the job with a delay determined by <code>vtk_job_control -delay option</code>.</p>
<code>NC_TEST_KERBEROS</code>	<code>nc kerberos</code>	Used only for testing. If set to any value, then the kerberos code assumes that the Kerberos Credential Cache file is good instead of testing it with <code>klist</code> . Adds verbosity to <code>vovbytepump</code> (only useful to developers).
<code>NC_URL</code>	<code>nc run</code>	<p>This is a useful variable to speedup job submission with <code>nc run</code>. If this variable is set, use its value to compute the URL for the submitted job instead of asking the server. This saves one round-trip from client to server. Also consider <code>NC_DEFAULT_JOBCLASS</code> and <code>NC_VALID_DIRECTORIES</code></p> <p>Example:</p> <pre>setenv NC_URL `vovbrowser -url ""`</pre>
<code>NC_VALID_DIRECTORIES</code>	<code>nc run</code>	<p>This is a useful variable to speedup job submission with <code>nc run</code>. If this variable is set to a space-separated list of directories, we use its value to determine valid submission directories. This is supposed to be the same as the property <code>NC_VALID_DIRECTORIES</code></p>

Variable Name	Used By	Description
		<p>attached to Object 1. Also consider NC_URL and NC_DEFAULT_JOBCLASS.</p> <p>Example 1: <code>setenv NC_VALID_DIRECTORIES "/scratch/ /remote/work"</code></p> <p>Example 2: <code>setenv NC_VALID_DIRECTORIES ""</code></p>
<i>PATH</i>		Normally defined in the standard environment (POSIX).
<i>RLM_PROJECT</i>	nc	This variable is used by Reprise to specify a project for a checkout. It is also used by <code>nc run</code> to define the default value of the <code>jobproj</code> field of a job. See also VOV_JOBPROJ and LM_PROJECT.
<i>SECURITY_LEVEL</i>	vovsh	<p>If set, this variable can lower the security level of the vovsh. This variable can never increase the security level, which is determined by <code>security.tcl</code>. The variable is an integer. The common values are</p> <ul style="list-style-type: none"> • 2 = READONLY • 3 = USER • 4 = LEADER <pre>% env SECURITY_LEVEL=3 vovsh -f myscript.tcl</pre>
<i>SHELL</i>	ves vovprompt vovsetupuser	Normally defined in the standard environment (POSIX).
<i>TEMP</i>	vovbytepump VovTmpFile	Normally defined in Windows to point to a writable directory for temporary files, this variable is used also on UNIX in the Tcl procedure <code>VovTmpFile</code> .
<i>TZ</i>		Normally defined in the standard environment (POSIX).
<i>USER</i>		Normally defined in the standard environment (POSIX).
<i>VNCSWD</i>	nc ncmgr	Normally used to specify the server working directory for Accelerator. Its value is normally <code>\$VOVDIR/../../vnc</code> . For installations including a mix of UNIX and Windows NT, the value of <code>VNCSWD</code> is normally set also in <code>\$VOVDIR/local/vovinit.bat</code> which is executed as part of the initialization sequence on Windows NT.

Variable Name	Used By	Description
VOVARCH		Identifies the application architecture. It is computed by the tool <code>vovarch</code> . It is normally set in <code>\$VOVDIR/etc/vovrc.{sh, csh, bat}</code> .
VOVBUILDOPTIONS	vovmake	Pass options to be used by <code>vovbuild</code> inside of <code>vovmake</code> .
VOVCLEARCASE		If set to "ENABLED", this variable activates the ClearCase routines to track dependencies on both the "view extended name" and the "version extended name" of a file.
VOVCONSOLE_SUBMIT	vovconsole	<p>Define that environment variable so that the <code>vovconsole</code> automatically <code>bsub</code>s itself. Typical usage:</p> <pre>setenv VOVCONSOLE_SUBMIT_CMD 'bsub -o /dev/null -R "rusage[mem=200]" '</pre> <p>Use:</p> <pre>printenv grep VOVCONSOLE_SUBMIT_CMD VOVCONSOLE_SUBMIT_CMD=bsub -R "rusage[mem=200]"</pre> <p>to check the value of that variable as <code>echo \$VOVCONSOLE_SUBMIT_CMD</code> would return a No match error.</p> <p>You may also want to use:</p> <pre>setenv VOVCONSOLE_SUBMIT_CMD 'bsub -R "rusage[mem=200]" -I '</pre> <p>to debug submission errors.</p>
VOVDIR		The root of the VOV distribution tree for a specific architecture. <code>VOVDIR</code> always includes the architecture. From <code>\$VOVDIR</code> , you can get the binaries in <code>\$VOVDIR/bin</code> .
VOVEQUIV_CACHE_FILE		Provides the ability to override the default behavior of using a server-side cache of path equivalence information. Set to a value of "legacy" to instruct clients to read the file directly. This is a legacy method that requires that all clients have access to the server working directory so they can parse <code>equiv.tcl</code> file for entries, and read/write access to the <code>equiv.caches</code> directory so the entries can be written to a host-based cache file for future use. Set to a custom cache file path to instruct clients to read a specific cache file only.

Variable Name	Used By	Description
		This is a special method used in corner cases where directories may not be the same but should be forced to be considered the same. This is utilized mainly by Monitor agent single-file distributables.
VOVGMAKE	vovmake	Specify the path to gmaek_with_vov_extension
VOVLSFINFOD	vsm nc mon	This variable should have the form HOST:PORT and represents the way to access the vovlsfinfod daemon. The PORT is normally 6006.
VOVLSF_USE_NC	LSF emulation utilities	Set to 0 to prevent LSF emulation from interacting with Accelerator. This is only required when wanting LSF emulation to interact with a FlowTracer server instead of Accelerator.
VOVMAKEDOCLEANUP	vovmake	If set to 1, do not cleanup the intermediate Flow file caused by dumping the makefiles rules.
VOVPROJECT_SUBMIT_	vovproject vovserver	<p>This variable contains the command used by vovproject to launch the vovserver, for example using bsub. Typical usage:</p> <pre>setenv VOVPROJECT_SUBMIT_CMD 'bsub -o logs/ server.20130531_123000.log -R "rusage[mem=200]" '</pre> <p>Since this variable often has a value with tricky characters, we suggest you use:</p> <pre>% printenv grep VOVPROJECT_SUBMIT_CMD VOVPROJECT_SUBMIT_CMD=bsub -R "rusage[mem=200]"</pre> <p>to check the value of that variable as echo \$VOVPROJECT_SUBMIT_CMD would return a No match error.</p> <p>For debugging purposes, you may also want to use interactive options like the option -I in this example:</p> <pre>% setenv VOVPROJECT_SUBMIT_CMD 'bsub -R "rusage[mem=200]" -I '</pre>
VOVRTILIB		Control the name of the dynamic library used for Runtime Interception. By default, the library is \$VOVDIR/lib/vovrti.so. This variable is used mostly by developers.

Variable Name	Used By	Description
<code>VOVRTILOGFILE</code>		Specify location of log file for debugging information about the behavior of VOVRTI.
<code>VOVSAVEPROMPT</code>	<code>vovprompt</code> <code>ves</code> <code>veprestore</code>	Used to store the original value of the prompt.
<code>VOVSETUPFILE</code>	<code>indirect taskers</code> <code>nc run ID ...</code>	If this variable is set, then the indirect tasker use its value when submitting a job from FlowTracer to Accelerator. Specifically, the value of <code>VOVSETUPFILE</code> becomes the parameter for the <code>PROJECT</code> environment. In other words, the indirect tasker submits a job of type <code>vovrun</code> with environment " <code>PROJECT(\$VOVSETUPFILE)</code> "
<code>VOVTMPVAR</code>	<code>vovtasker</code>	Temporary variable used only in the initialization of the multi-environment subsystem. Do not use this variable because it is likely to be over-written.
<code>VOV_32BIT</code>	<code>vovarch</code>	See <code>VOV_BIT_MODE</code> instead.
<code>VOV_ALARM</code>	<code>vovgetgroups</code>	Used by <code>vovgetgroups</code> to avoid hanging indefinitely. The default value is 10 seconds, but there is no maximum upper limit.
<code>VOV_BACKGROUND</code>		Sets the background color of the GUI and of all the browser pages. If not set, the color is chosen automatically by hashing the project name so that each project has a slightly different background. The background should be light. Examples: <div> <pre>setenv VOV_BACKGROUND wheat setenv VOV_BACKGROUND #FFB094</pre> </div>
<code>VOV_BIT_MODE</code>	<code>vovarch</code>	This variable is used to select which architecture to use on 64-bit Linux machines and affects the output of the <code>vovarch</code> command. By default the command returns "linux64" on 64-bit Linux machines. However, these machines can also run 32-bit binaries as long as the standard 32-bit libraries are included in the Linux installation. To force a 64-bit machine to run Altair Accelerator in 32-bit mode, set this environment

Variable Name	Used By	Description
		variable to "linux" before sourcing the shell-appropriate vovrc setup script. Example: <pre>setenv VOV_BIT_MODE "linux64"</pre>
VOV_BJOBS_JOBID_WIDTH	bjobs	Used to override the default job ID column width of -7. Characters to the left of the width limit are truncated and replaced with an asterisk to indicate the truncation. Pass a negative integer for left justification. Pass a positive integer for right justification.
VOV_CAPSULE_DIR	vov vw vrt vw2 vtrace	Specify a list of directories where capsules can be found. The list is separated by a colon ':' on Unix and by a semicolon ';' on Windows. The other directories that are searched are: <ul style="list-style-type: none"> • The current working directory • \$VOV_CAPSULE_DIR • \$VOVDIR/local/capsules • \$VOVDIR/tcl/vtcl/capsules
VOV_CGROUPS_ENABLE		If this variable is set to 1 in the environment of vovtaskerroot, then the CGROUP:RAM capability is enabled. <pre># Recommended in vnc.swd/setup.tcl setenv VOV_CGROUPS_ENABLE 1</pre>
VOV_CHANNEL	VIL	Full path to the vovchannel binary. Defaults to \$VOVDIR/bin/vovchannel.
VOV_CHANNEL_OPEN	VIL vovchannel.d	Used internally by VIL and by vovchannel. Should not be set by the user.
VOV_CLIENT_NAME		If set, this variable is used to set the name field for a client. The client name is only visible from the /admin? page=clients browser page.
VOV_CONFLICT_CONTINUE	vw vrt vov vtrace	If there is a conflict, check the value of this variable. If it is ABORT or CONTINUE, behave correspondingly. If it has another value, print a warning and return ABORT. If the variable is not set, in case of conflict a dialog will be posted to determine what to do. This variable is used to control Runtime Tracing in case of conflict. If the behavior is ABORT, then the tool is aborted at the

Variable Name	Used By	Description
	vw2	first conflict. If the behavior is CONTINUE, then the tool continues to execute.
VOV_DB_APP		This variable can be used to select different database configurations for the same VOV project. In the file *.swd/db/config.tcl, use VovSQL::getDbApp to return the value of this variable.
VOV_DEBUG_BYTEPUMP	vovbytepump	Adds verbosity to vovbytepump (only useful to developers).
VOV_DEBUG_ENV	vel ves	Adds verbosity to vel and ves. Set the value to 1 for debugging, to 0 to disable debugging.
VOV_DEBUG_EQUIV	vw2 vov vrt vtrace vw vovtasker vovtaskerroot	Adds verbosity to equivalence handling module.
VOV_DEBUG_FLAGS	All VOV clients	<p>This variable is to be used only by developers and experts. It controls debugging output to monitor communication between clients and server. It also controls verbosity of some modules. Here is a quick reminder of the possible values for this variable.</p> <ul style="list-style-type: none"> • 1 - CHANNEL • 2 - CLIENT • 4 - SERVER • 8 - VIL • 16 - RPC Codes • 128 - Exclusions • 256 - VOVRTI <p>The definition of the flags is documented in \\${VOVDIR}/src/vil/vil.h</p>
VOV_DEBUG_GETGROUPS	vovtasker	Adds verbosity to vovgetgroups.
VOV_DEBUG_LDAPAUTH	vovserver	Adds verbosity to LDAP authentication procedure.

Variable Name	Used By	Description
	vovldapauth	
VOV_DEBUG_LIMITS	vw2 vov vrt vtrace vw vovtasker vovtaskerroot	Adds verbosity to tools that set the resource limits like <code>cputime</code> and <code>coredumpsize</code> . Affects <code>vovtasker</code> and <code>vw2</code> .
VOV_DEBUG_MD5	vw vov vrt vw2	Adds verbosity to wrapper code used to implement MD5 based barriers. Any value of this variable activates the verbosity. Unset the variable to disable.
VOV_DEBUG_NO_START	vovtasker	Provides debug info for jobs launched by taskers that do not start successfully. Causes the tasker to run the <code>vov_diagnostic_no_start</code> script to generate detailed debug information on why a job did not start. Set to 1 for coarse info (most common setting). Set to 2 for more detail (should only be done for brief periods on selected taskers due to high server load).
VOV_DEBUG_ORIGARGS	nc	Enables the capturing of the original argument list to <code>nc run</code> and stores it as a property (ORIGARGS) for debugging purposes. Set this variable to 1 in the submit shell or NC <code>setup.tcl</code> file to enable. Do not use this function unless it is needed, however, since it will cause the server to consume a significant amount of RAM with higher workloads.
VOV_DEBUG_PAMAUTH	vovserver vovpamauth	Adds verbosity to pam authentication procedure.
VOV_DEBUG_PTY	vovtasker vovsh	Adds verbosity to PTY code, used in <code>vovsh</code> and <code>vovtasker</code> with interactive jobs that require a PTY. Set the value to 1 for basic information, or to 2 to see every character being sent.
VOV_DEBUG_TASKER_C	vovtasker	Used to troubleshoot problems with subtasker changing to the run directory. Only useful for debugging, should

Variable Name	Used By	Description
	vovtaskerroot	normally be unset. Debugging output appears in vovtasker's log file. When set, activates a call the script <code>vov_diagnostics_chdir</code> whenever a vovtasker is unable to start a job because the working directory for the job cannot be reached. This type of errors can be hard to diagnose, for example in cases when the error is caused by slow or overloaded auto-mounters.
<code>VOV_DEBUG_TASKER_C</code>	vovtasker vovtaskerroot	Adds verbosity to the code used to compute os groups for users.
<code>VOV_DEBUG_TASKER_N</code>	vovtasker	Adds verbosity to the indirect NC tasker. Set variable to 1 to enable, to 0 to disable. Default is 0.
<code>VOV_DEBUG_SYMLINKS</code>	vw vrt vov	Adds verbosity to code used to expand symlinks. Often used in conjunction with <code>VOV_SYNC_CACHE_DIR</code> Set the value to 1 to activate, unset to deactivate.
<code>VOV_DEBUG_VTKGRAPH</code>	vovsh GUI	Adds verbosity to graph widget in <code>vovsh</code> .
<code>VOV_DELAY_BEGIN</code>	vw	Delay job execution after start by specified number of seconds. Used to compensate for offset in filesystem clock.
<code>VOV_DELAY_END</code>	vw	Introduce a small delay after the job is done and before checking the timestamps of the outputs.
<code>VOV_DELAY_FIRE</code>	vw	Delay tasker firing by specified number of seconds. Used to compensate for offset in filesystem clock.
<code>VOV_DISABLE_DST_CH</code>		Disable the code used to compensate for "Daylight Saving Time" overcompensation bug on Windows NT. By default, the code is active.
<code>VOV_DISABLE_SHARED</code>	vovtaskerroot vovtasker	Instructs the tasker (the top level vovtaskerroot/ vovtasker process) to disable the calculation of the job's CURRAM usage by extracting Pss usage statistics from <code>/proc/PID/smmaps</code> system files. In disabled mode, the RSS usage for all processes in a job will be used to calculate CURRAM instead. Specify 0 to enable and 1 to disable. Default: 0
<code>VOV_DISABLE_VIL_RE</code>	VIL	Disable redirection of stdout and stderr through <code>VIL</code> and <code>vovchannel</code> .

Variable Name	Used By	Description
	vovchannel	
VOV_EDITOR	vovsh GUI	Used by the GUI to decide which editor to use. If not set, it defaults to <code>write</code> on Win64 and to <code>xterm -e vi</code> on UNIX.
VOV_ENV	ves vw	Used to support multiple environments. Stores the name of the current environment. If it is not set, the environment name is assumed to be DEFAULT and the multiple environment support is disabled.
VOV_ENV_DIR	vel ves vovtasker	Used to support multiple environments. Its value is a list of the directories that contain the environment scripts. The list elements are separated by colons ':' on UNIX and by semicolons ';' on Windows NT.
VOV_ENV_SOURCE	vwrap	Used by <code>vwrap</code> to source the environment snapshot. Should not be set by the user.
VOV_ENV_STRICT	ves	By default, <code>ves</code> ignores errors when switching environments. When this variable is set to a non-zero value, any environment switch which writes to <code>stderr</code> will be strictly treated as an error and the environment switch will fail.
VOV_EXCLUDE_FILES	vw vov vrt vw1	List of files containing exclusion rules. If this variable is not set, the wrappers only read the <code>exclude.tcl</code> file in the SWD directory. Missing files in the list are silently ignored. Errors in sourcing the files are also silently ignored. The files in the list are separated by colons ':' on UNIX and by semicolons ';' on Windows.
VOV_FAIL_ON_CONFLI		Use VOV_CONFLICT_CONTROL instead. If there is a conflict, cause the job to fail right away. Do not pop-up a dialog or wait for user input. Especially useful in the environment for taskers in Windows-NT.
VOV_FDL_ONLY	viltool VovInput VovOutput VovFdl VovResources	If this variable is set, the VIL-Tools do not communicate with the server but instead generate the equivalent FDL code. If the value of the variable is "1", the code is printed on <code>stdout</code> . If the value of the variable is the name of a writable file, the code is written to that file (no locking). A value of "0" is the same as not having the variable defined. See example in <code>\$VOVDIR/training/basic/example.csh</code> .

Variable Name	Used By	Description
<code>VOV_FIFO_OPEN</code>	VIL vovchannel	Used internally by VIL and by vovchannel. Should not be set by the user.
<code>VOV_FOREGROUND</code>	vovserver vovsh GUI	Sets the foreground color of the VOV dialogs. If not set, the color is chosen automatically so that each project has a slightly different foreground. The foreground should be dark. Examples: <div> <pre>setenv VOV_FOREGROUND black setenv VOV_FOREGROUND #00322F</pre> </div>
<code>VOV_FT_LOGS</code>	vovrun	When using vovrun (see "indirect taskers") you can choose to propagate the list of log files from FT to NC. This is done by setting VOV_FT_LOGS to 1 in the environment for vovrun.
<code>VOV_GOTOHOST_COMMAND</code>	vovconsole	Command used in the "GoToHost" callback in vovconsole. The command should contain the token '@HOST@' which will be replaced with the name of the host. The default value is "vovxrch @HOST@".
<code>VOV_GRABBED_RESOURCES</code>		When a job is being executed on a vovtasker this environment variable is set to describe the resources that have been grabbed for this job.
<code>VOV_GROUP</code>		If present, it defines the FairShare group for jobs.
<code>VOV_GROUP_PURGE_PERIOD</code>	vovtaskerroot vovtasker	Instructs the tasker to periodically purge cached UNIX group information, including gid/name maps and user group lists. Specified in seconds (minimum 60). Default: 0 (disabled).
<code>VOV_HOST_HTTP_NAME</code>		Variable to control the name of the server as it is to be used in the HTTP interface. If not defined, use VOV_HOST_NAME.
<code>VOV_HOST_IP</code>		If present, used to connect to the vovserver. You may want to use this variable to avoid calling the function <code>getaddrinfo()</code> which can be expensive on systems in which the DNS is not working properly.

Variable Name	Used By	Description
		<p>Warning: For some values of this variable, the clients will hang indefinitely. The recommendation, in general, is to not use this variable. The variable overrides the value of VOV_HOST_NAME. Example:</p> <pre>% setenv VOV_HOST_IP 192.168.33.12</pre>
VOV_HOST_NAME		Used by all tools. It is the name of the machine on which the vovserver is running, unless VOV_HOST_IP is also set, in which case it takes precedence.
VOV_HTTP_UNSECURE	vovserver	Disables authentication in the browser. Everyone can connect, but their security level will be READONLY. Used only by vovserver. Its usage is discouraged. Use <code>config(httpSecure)</code> in <code>policy.tcl</code> file instead.
VOV_INSTANCE_CHARGE_BOUNDARY	vovtasker vovtaskerroot	This variable is used mostly for testing, so you don't have to wait for a full hour to test the effect of the other variable VOV_INSTANCE_LAUNCH_TS. If not defined, the CHARGE BOUNDARY is considered to be 3600, which is 1 hour in seconds. The minimum value for this variable is 300 seconds. There is no upper bound. For testing, this variable can be set to lower values, like 300. Recommendation: do not use this variable and accept the default value.
VOV_INSTANCE_LAUNCH_TS	vovtasker vovtaskerroot	Used in cloud computing environments where the charges per instance are rounded up to the nearest hour. If set, it represents the launch time of the instance. If the vovtasker/vovtaskerroot uses the option <code>-z TIMESPEC</code> (e.g. <code>-z 2m</code>), the idea is to terminate the vovtasker after it has been idle for the specified amount. However, if the variable VOV_INSTANCE_LAUNCH_TS is set, then the tasker is kept running if more than 4 minutes away from the hour boundary. This is to avoid wasting a vovtasker that is essentially already paid for. The ordinary way to set this variable is in the boot script for an instance using the value returned, for example, by <code>vov_ec2_get_launch_time</code>

Variable Name	Used By	Description
<code>VOV_INTERACTIVE_AUTH_TIMEOUT</code>	vovtasker	If interactive jobs are failing due to authentication timeouts on the vovtasker, this environment variable can be used to increase the timeout. For example, <code>setenv VOV_INTERACTIVE_AUTH_TIMEOUT 5s</code> will set the authentication timeout used by interactive jobs to 5 seconds. The default is 2s. The value of <code>VOV_INTERACTIVE_AUTH_TIMEOUT</code> is silently restricted between 2s and 30s. It is recommended to use as small a value as feasible since this also blocks the vovtasker from running other jobs. Too long a timeout can also lead to the vovserver marking the vovtasker as sick.
<code>VOV_INTERACTIVE_PING</code>	nc run -I	For sites that have firewalls that close inactive connections, you can use this variable to force Accelerator to keep the connection alive when running in interactive mode (nc -I, etc.). For example: <pre>setenv VOV_INTERACTIVE_PING 2m</pre> will send a message every 20m to the interactive job. The value of <code>VOV_INTERACTIVE_PING</code> is silently limited to a minimum of 1m.
<code>VOV_JOBCLASS</code>	vw2 vw vov vrt	If a new job is being added to the trace by direct execution and this variable is set, it will be used as "jobclass" of the job.
<code>VOV_JOBCLASS_DIRS</code>	nc FDL vovresourced	Used to specify jobclass directories in addition to the system ones in <code>vnc.swd/jobclass</code> , <code>\\$VOVDIR/local/jobclass</code> and <code>\\$VOVDIR/etc/jobclass</code> . Its value is a list of the directories to search for jobclass files. The list elements are separated by colons ':' on UNIX and by semicolons ';' on Windows.
<code>VOV_JOB_COUNTER</code>	vovcounters vlmstat vtool	In the form <code>PORT@HOST,PROJNAME</code> , points to the license emulation server. Examples: <pre>% setenv VOV_JOB_COUNTER 6306@somehost,licadm % setenv VOV_JOB_COUNTER 5555@otherhost,licmon</pre>
<code>VOV_JOB_COUNTER_PROXY</code>	vovcounters	Not used.

Variable Name	Used By	Description
VOV_JOB_COUNTER_VERBOSE	vovcounters	Make vovcounters more verbose.
VOV_JOBID		When a job is running, it can find its own ID by querying this variable.
VOV_JOBINDEX		When a job array is submitted, the VOV_JOBINDEX environment variable will be set in the execution environment of each job in the array. The variable is for consumption only and is not intended to be set by the user at any time.
VOV_JOBNAME	vw2 vw vov vrt	If a new job is being added to the trace by direct execution and this variable is set, it will be used as "jobname" of the job.
VOV_JOBPROJ	nc	Used by nc run to define the default value of the jobproj field of a job. The order in which these variables are checked is VOV_JOBPROJ LM_PROJECT RLM_PROJECT and the first one found prevails.
VOV_JOBSLOT		Set only for jobs running on a tasker. It represents the slot number in which the job is running.
VOV_JOB_BLOCKNAME	vovsql_record_one_job	Allows the CAD engineers to define a name for a block. Examples are: CPU, Cache, Sequencer, Decoder, Bus, PCI, DRAM1... If it is not defined, vovsql_record_one_job will also look in the environment for a value from BLOCK.
VOV_JOB_BLOCKRELEASE	vovsql_record_one_job	Complement/qualifier to VOV_JOB_BLOCKNAME. It allows the CAD engineer to record what specific release of the block is currently being used. For one NAME and one SPIN, there maybe several RELEASEs.
VOV_JOB_BLOCKSPIN	vovsql_record_one_job	Complement/qualifier to VOV_JOB_BLOCKNAME. It allows the CAD engineer to record what specific spin of the block is being used. One NAME will typically have multiple SPINs. In electronic design, it is usually the floorplan name.
VOV_JOB_BLOCKVARIANT	vovsql_record_one_job	Complement/qualifier to VOV_JOB_BLOCKNAME. It allows the CAD engineer to record what specific variant of the block is being used. A VARIANT is typically of larger scope than a NAME. One VARIANT will typically

Variable Name	Used By	Description
		have multiple NAMES. It's like a family or generation of blocks.
VOV_JOB_EXPERIMENT	vovsql_record_one_job	Define a short name for the experiment in progress. Examples: congestion_0p60 or congestion_0p65, or syn_effort_low, or 16_tracks, etc. If not defined, the script will look for a common environment variable: NICKNAME. The experiment name is picked by each designer as they go through their design cycle and is meaningful mostly to the designer.
VOV_JOB_EXPERIMENT	vovsql_record_one_job	Define a root directory for the experiment in progress. If not defined, but an experiment config file was found, the location of the experiment config file will be used.
VOV_JOB_FLOWNAME	vovsql_record_one_job	Allows the CAD engineers to define a name for a flow. Examples are: synthesis, verification, placenoute.
VOV_JOB_FLOWRELEASE	vovsql_record_one_job	Complement/qualifier to VOV_JOB_FLOWNAME. It allows the CAD engineer to record what specific release of the flow is being used. For one NAME and one SPIN, there maybe several RELEASEs. Examples are: build_78666, rel_1.0, 2012.09...
VOV_JOB_FLOWSPIN	vovsql_record_one_job	Complement/qualifier to VOV_JOB_FLOWNAME. It allows the CAD engineer to record what specific spin of the flow is being used. One NAME will typically have multiple SPINs. It could be considered as the personality of the flow. Examples are: Q1_stretch...
VOV_JOB_FLOWVARIANT	vovsql_record_one_job	Complement/qualifier to VOV_JOB_FLOWNAME. It allows the CAD engineer to record what specific variant of the flow is being used. A VARIANT is typically of larger scope than a NAME. One VARIANT will typically have multiple NAMES. It's like a family of flows. Examples are: FullChip, Block...
VOV_JOB_NAME	vovsql_record_one_job	Should be set to the name of the job (see the "activity" paragraph in the "recording jobs" page for a discussion on job names). If it is not defined, the jobname defined during the building of the graph will be used, and otherwise an abbreviated command line will be used.
VOV_JOB_PREDICTION	vovsql_record_one_job	Name of the directory where to generate the memory prediction files. Used by gen_predict_files .

Variable Name	Used By	Description
VOV_JOB_PREDICTNAME	vovsql_record_one_job	Enables invisible override of VOV_JOB_NAME so that a different name can be used for the job just for the matters of memory usage prediction. If it does not exist, the value of VOV_JOB_NAME is used, and if VOV_JOB_NAME does not exist, then the jobname defined during the building of the graph will be used, and if that does not exist, the tool name will be used.
VOV_JOB_PROJECTNAME	vovsql_record_one_job	Allows the CAD engineers to define a name for a project. Examples are: SandyBridge, IvyBridge, Nehalem, Westmere, Bulldozer, Bobcat, Redwood, Cedar, Krait... If it is not defined, vovsql_record_one_job will also look in the environment for a value from LSB_PROJECT, then LM_PROJECT, then PROJECT.
VOV_JOB_PROJECTRELEASE	vovsql_record_one_job	Complement/qualifier to VOV_JOB_PROJECTNAME. It allows the CAD engineer to record what specific release of the project is currently being used. For one NAME and one SPIN, there maybe several RELEASEs. Examples are: rel_1.0, nl-2.1, fp10... If it is not defined, vovsql_record_one_job will also look in the environment for a value from RELEASE.
VOV_JOB_PROJECTSPIN	vovsql_record_one_job	Complement/qualifier to VOV_JOB_PROJECTNAME. It allows the CAD engineer to record what specific spin of the project is being used. One NAME will typically have multiple SPINs. In electronic design, it is usually the tapeout name. Examples are: v1, sa11... If it is not defined, vovsql_record_one_job will also look in the environment for a value from SPIN then TAPEOUT.
VOV_JOB_PROJECTVARIANT	vovsql_record_one_job	Complement/qualifier to VOV_JOB_PROJECTNAME. It allows the CAD engineer to record what specific variant of the project is being used. A VARIANT is typically of larger scope than a NAME. One VARIANT will typically have multiple NAMES. It's like a family or generation of projects. Examples are: Evergreen, SouthernIslands, Kepler... If it is not defined, vovsql_record_one_job will also look in the environment for a value from VARIANT.
VOV_JOB_SITECITY	vovsql_record_one_job	Provide a city name for a site. Some companies use a convention for site names that may not be obvious to a newcomer. Example: LSDC would stand for Lone Star Design Centre. If you're not American, you may

Variable Name	Used By	Description
		not know this means Austin, Texas. The SITECITY field lets you record the name of the city where the site is. Example: Austin.
VOV_JOB_SITECOUNTRY	vovsql_record_one_job	Records the country of the current site.
VOV_JOB_SITENAME	vovsql_record_one_job	Provide a site name. This is usually used to record a common disk space / server farm. Examples are: Hyderabad, HYDC, LoneStar, sj, Canada. If it is not defined, vovsql_record_one_job will also look in the environment for a value from SITE, then XSITE.
VOV_JOB_SITETIMEZONE	vovsql_record_one_job	Records the timezone of the current site.
VOV_JOB_USERSITENAME	vovsql_record_one_job	User site vs host site. With global companies, a user located in Hyderabad, India may be working remotely on a server farm located in Anchorage, Alaska, USA. Due to the hierarchy of the company, you may want to capture the difference between the site where the user is physically located (Hyderabad) as (s)he may report through a local hierarchy, and the site where the job is running (Anchorage) as this cost depends from the American hierarchy. Depending on their level in the organization, some persons may be interested in activity reports based on people's locations (such as, what are my people in Hyderabad doing), while others may be interested in how is the Anchorage farm doing. The best way to address this is to use a background data acquisition script that populates/refreshes the users' sites everyday from a central location like LDAP or Microsoft ActiveDirectory, and to let the environment variables take care of the job/compute host's site. VOV_JOB_USERSITENAME can provide a different site name just for the user. This is usually used to record the physical location of the human running the jobs. Examples are: Hyderabad, HYDC, LoneStar, sj, Canada. If it is not defined, vovsql_record_one_job will also look in the environment for a value from VOV_JOB_SITENAME, then SITE, then XSITE.
VOV_LICENSE_KEY	vovserver vovreadlicutil	Specify a path for the VOV License Key file. The default is license.key in the PROJECT.swd directory.

Variable Name	Used By	Description
VOV_LICMON	ftlm_lmproject	<p>Register job project information. It is used to point to the instance of Monitor that captures checkout information. It has the following forms:</p> <pre> Non-SSL: <host>:<port> Example: lmsrv:5555 SSL: <host>:<port>:ssl Example: lmsrv:5555:ssl </pre>
VOV_LIMIT_coredumps	vw vovtasker	<p>This variable, together with other VOV_LIMIT_* variables, allows passing of 'limit' information to a job using the environment.</p> <p>Common use in Accelerator:</p> <pre>% nc run -e "BASE +D(VOV_LIMIT_coredumps=100M)" job_command</pre> <p>On Linux, this limit controls the system limit called RLIMIT_CORE.</p>
VOV_LIMIT_cputime	vw vovtasker	<p>Together with other VOV_LIMIT_* variables, allows passing of 'limit' information to a job using the environment. This one controls the "cputime" limit.</p> <p>Common use in Accelerator:</p> <pre>% nc run -e "BASE+D(VOV_LIMIT_cputime=10)" short_job</pre>
VOV_LIMIT_datasize	vw vovtasker	<p>Together with other VOV_LIMIT_* variables, allows passing of 'limit' information to a job using the environment. Common use in Accelerator:</p> <pre>% nc run -e "BASE +D(VOV_LIMIT_datasize=8192k)" job_command</pre> <p>The value of the variable must be an sequence of digits followed by an optional unit indicator letter (case insensitive), or the string value 'unlimited'. If there is no unit indicator, the units are kilobytes (1024 bytes). The recognized unit indicators are:</p> <ul style="list-style-type: none"> • k or K, kilobytes (same as no unit indicator) • M, Megabytes (multiply by 1024) • G, Gigabytes (multiply by 1024*1024) • T, Terabytes (multiply by 1024*1024*1024)


Variable Name	Used By	Description
<code>VOV_LIMIT_descriptvw</code>	<code>vw</code> <code>vovtasker</code>	Do not use this variable. Use <code>VOV_LIMIT_openfiles</code> instead.
<code>VOV_LIMIT_filesizevw</code>	<code>vw</code> <code>vovtasker</code>	<p>Together with other <code>VOV_LIMIT_*</code> variables, allows passing of 'limit' information to a job using the environment. Common use in Accelerator:</p> <pre>% nc run -e "BASE+D(VOV_LIMIT_filesize=100M)" job_command</pre> <p>On Linux, this limit controls the system limit called <code>RLIMIT_FSIZE</code>.</p> <p>The value of the variable must be an sequence of digits followed by an optional unit indicator letter (case insensitive), or the string value 'unlimited'. If there is no unit indicator, the units are kilobytes (1024 bytes). The recognized unit indicators are:</p> <ul style="list-style-type: none"> • k or K, kilobytes (same as no unit indicator) • M, Megabytes (multiply by 1024) • G, Gigabytes (multiply by 1024*1024) • T, Terabytes (multiply by 1024*1024*1024)
<code>VOV_LIMIT_maxprocvw</code>	<code>vw</code> <code>vovtasker</code>	<p>Together with other <code>VOV_LIMIT_*</code> variables, allows passing of 'limit' information to a job using the environment. Common use in Accelerator:</p> <pre>% nc run -e "BASE+D(VOV_LIMIT_maxproc=100)" job_command</pre> <p>On Linux, this limit controls the system limit called <code>RLIMIT_NPROC</code>.</p> <p>The value of the variable must be an integer.</p>
<code>VOV_LIMIT_memorylockvw</code>	<code>vw</code> <code>vovtasker</code>	<p>Together with other <code>VOV_LIMIT_*</code> variables, allows passing of 'limit' information to a job using the environment. Common use in Accelerator:</p> <pre>% nc run -e "BASE +D(VOV_LIMIT_memorylocked=200)" job_command</pre> <p>On Linux, this limit controls the system limit called <code>RLIMIT_MEMLOCK</code>.</p> <p>The value of the variable must be an sequence of digits followed by an optional unit indicator letter (case</p>

Variable Name	Used By	Description
		<p>insensitive), or the string value 'unlimited'. If there is no unit indicator, the units are kilobytes (1024 bytes). The recognized unit indicators are:</p> <ul style="list-style-type: none"> • k or K, kilobytes (same as no unit indicator) • M, Megabytes (multiply by 1024) • G, Gigabytes (multiply by 1024*1024) • T, Terabytes (multiply by 1024*1024*1024)
VOV_LIMIT_memoryuse	vw vovtasker	<p>This variable is a no-op and is provided for backward compatibility only. The same is true for the memoryuse limit in UNIX shells as well.</p>
VOV_LIMIT_openfiles	vw vovtasker	<p>Together with other VOV_LIMIT_* variables, allows passing of 'limit' information to a job using the environment. This one controls the number of file descriptors that can be used by a job. Common use in Accelerator:</p> <pre>% nc run -e "BASE +D(VOV_LIMIT_openfiles=1024)" job_command</pre> <p>In C-shell, this limit is also called 'descriptors'.</p>
VOV_LIMIT_stacksize	vw vovtasker	<p>Together with other VOV_LIMIT_* variables, allows passing of 'limit' information to a job using the environment. Common use in Accelerator:</p> <pre>% nc run -e "BASE +D(VOV_LIMIT_stacksize=8192k)" job_command</pre> <p>The value of the variable must be an sequence of digits followed by an optional unit indicator letter (case insensitive), or the string value 'unlimited'. If there is no unit indicator, the units are kilobytes (1024 bytes). The recognized unit indicators are:</p> <ul style="list-style-type: none"> • k or K, kilobytes (same as no unit indicator) • M, Megabytes (multiply by 1024) • G, Gigabytes (multiply by 1024*1024) • T, Terabytes (multiply by 1024*1024*1024)

Variable Name	Used By	Description
<code>VOV_LIMIT_vmemoryuse</code>	<code>vw</code> <code>vovtasker</code>	<p>Together with other <code>VOV_LIMIT_*</code> variables, is used to pass shell 'limit' information to a job via the environment. Common use in Accelerator:</p> <pre>% nc run -e "BASE +D(VOV_LIMIT_vmemoryuse=100M)" job_command</pre> <p>On Linux, this limit controls the system limit called <code>RLIMIT_AS</code> (address space).</p> <p>The value of the variable must be an sequence of digits followed by an optional unit indicator letter (case insensitive), or the string value 'unlimited'. If there is no unit indicator, the units are kilobytes (1024 bytes). The recognized unit indicators are:</p> <ul style="list-style-type: none"> • k or K, kilobytes (same as no unit indicator) • M, Megabytes (multiply by 1024) • G, Gigabytes (multiply by 1024*1024) • T, Terabytes (multiply by 1024*1024*1024)
<code>VOV_LIVERECORDER_DIRECTORY</code>	<code>vovsh</code>	Specify directory for LiveRecorder log file for <code>vovsh</code> . Default is <code>/tmp</code> .
<code>VOV_LIVERECORDER_SIZE</code>	<code>vovsh</code>	Specify maximum size (MB) of LiveRecorder log file for <code>vovsh</code> . Default is 256.
<code>VOV_LIVERECORDER_ENABLED</code>	<code>vovsh</code>	Set to any value to enable LiveRecorder for <code>vovsh</code> .
<code>VOV_LMPROJECT_EXPIRATION</code>	<code>ftlm_lmproject</code>	When set, this overrides the default 4h expiration of the management project designation entered by the <code>ftlm_lmproject</code> script. The value is a VOV timespec, e.g. 7d is seven days.
<code>VOV_LM_BATCH</code>	<code>ftlm_batch_report</code>	Affects the generation of HTML for batch reports. In particular, if the variable is set (to any value), it disables the INPUT on images.
<code>VOV_LM_VARNAME</code>	<code>vw</code>	Specify a comma-separated list of variables to be set automatically on the basis of the grabbed resources. This variable is honored by the VOV wrappers (<code>vw</code> , <code>vov</code> , ...) so jobs that use no wrapper (like interactive jobs) do not have access to this service. An alternative method to set the <code>LM_LICENSE_FILE</code> automatically is provided by the utility <code>vovgetflexlm</code> daemons.

Variable Name	Used By	Description
<code>VOV_LOCAL_DIR</code>	<code>vovworkflowsim</code>	A local directory on the machine that can be used for I/O intensive operations. This is used by <code>vovworkflowsim</code> to store the intermediate results of the simulation. In that context, the default value of the directory is <code>/tmp/vov/wa</code> .
<code>VOV_LOG_LSFEMUL</code>	<code>bsub</code> <code>bjob</code> <code>bkill</code> <code>lsid</code> <code>lshosts</code> <code>bqueues</code> <code>bmgroup</code>	Choose file to be used to log the invocations of the LSF emulation scripts. Example: <code>setenv VOV_LOG_LSFEMUL ~/lsfemul.log</code>
<code>VOV_LSF_QUEUES</code>	<code>vsm</code> <code>nc mon</code>	Only needed when using indirect taskers from VOV to LSF. It represents a space-separated list of LSF queues to be monitored.
<code>VOV_MAX_WAIT_AFTER</code>	<code>vovtaskerroot</code> <code>vovtasker</code>	Deprecated. Please use <code>VOV_MAX_WAIT_TO_RECONNECT</code> instead.
<code>VOV_MAX_WAIT_NO_START</code>	<code>vovtaskerroot</code> <code>vovtasker</code>	Controls the time a tasker waits for <code>fork()/exec()</code> to work. This is called the "time to start" and it is typically a few milliseconds, but could grow large or infinite if there are problems with the machines. The valid range is 20 seconds to 8 days. The default value is 1 minute. This value may be a VOV timespec, e.g. 5m (5 minutes). This variable must be set when <code>vovtasker</code> starts, usually by setting it in the project <code>SWD/setup.tcl</code> file. The value may be changed at runtime with <code>vovtaskermgr configure TASKERNAME maxwaitnostart TIMESPEC</code> .
<code>VOV_MAX_WAIT_TO_RECONNECT</code>	<code>vovtaskerroot</code> <code>vovtasker</code>	Controls the time a tasker waits for a new <code>vovserver</code> to appear after losing connection to the server. The default value is 3 minutes for FlowTracer and 4 days for all other products. This variable must be set when the tasker starts.
<code>VOV_MD5_CMD</code>	<code>vw</code> <code>vov</code>	Change the command used to compute the md5 digest for purpose of computing MD5 based barriers. The

Variable Name	Used By	Description
	vrt vw2	default is <code>/sbin/md5 -q</code> on MacOSX and <code>/usr/bin/md5sum</code> on the other platforms.
<code>VOV_MIN_DESCRIPTOR</code>	ncmgr wxmgr	Specify the minimum number of file descriptors to be required by the server. The shell in which the server is being started will be tested, and if the current file descriptor soft limit is less than the specified minimum, an attempt will be made to raise the limit. If the specified minimum exceeds the system's file descriptor hard limit, an error will be displayed and the server will not be started. If not specified, the value used is 2500.
<code>VOV_NAME_SERVICE_RETRY</code>	vovserver vovtasker	Number of attempts the tasker will make to determine the current username via local name services. Retries may be necessitated by misconfigured or malfunctioning name services on the host. The default value is 3.
<code>VOV_NAME_SERVICE_WAIT</code>	vovserver vovtasker	Number of seconds to wait between retries when querying the name services for the current username. Retries may be necessitated by misconfigured or malfunctioning name services on the host. The default value is 3.
<code>VOV_NISRETRYCT</code>	vovtasker	In systems where NIS is slow or unreliable, this variable controls how many times the <code>getgrnam()</code> call should be retried (default 5s). Its use is discouraged: fix NIS instead.
<code>VOV_NISRETRYWAIT</code>	vovtasker	In systems where NIS is slow or unreliable, this variable controls how long to wait between retries (default 10s). Its use is discouraged: fix NIS instead.
<code>VOV_PAUSE_CHILD_SIGNAL</code>	vovtasker	<p>Linux platforms only</p> <p>Controls the signal sent to child processes when a tasker enters the PAUSED state. By default, this signal will be SIGSTOP. If an unsupported value is entered, the variable will be ignored and the default behavior ("STOP") will occur.</p> <p>Supported values:</p> <ol style="list-style-type: none"> 1. STOP : send SIGSTOP to child processes. This is the default behavior. 2. TSTP : send SIGTSTP to child processes.

Variable Name	Used By	Description
		<div>  Note: Child processes are not guaranteed to be suspended by this signal. </div> <ol style="list-style-type: none"> 3. NONE : do not send a signal to child processes. 4. <empty string> : same as NONE. <p>Examples:</p> <pre>% setenv VOV_PAUSE_CHILD_SIGNAL "STOP" % setenv VOV_PAUSE_CHILD_SIGNAL "TSTP" % setenv VOV_PAUSE_CHILD_SIGNAL "NONE" % setenv VOV_PAUSE_CHILD_SIGNAL ""</pre>
VOV_PBS_JOB_HISTORY	qdel	Direct the PBS emulation environment to behave as though job history were enabled. User must set the value of this variable to an integer greater than zero to enable this behavior. Default is disabled.
VOV_PIPES_TO_CLOSE	VIL vovchannel	Used internally by VIL on Windows and by vovchannel. Should not be set by the user.
VOV_PORT_NUMBER	vovserver vovtasker vovsh	<p>Controls the port number to which vovserver will bind. The port is used by various clients to establish a connection with the server. The variable can have the following forms:</p> <ol style="list-style-type: none"> 1. A number: specify a static port. 2. A colon-separated list of numbers. 3. Two numbers separated by a + (e.g. 8000+20000) where the first number represents a base and the second number a range for the selection of a port based on a hash of the project name. In the example 8000+20000, the port will be selected in the range from 8000 to 28000. 4. The value "automatic" (default), which maps to a integer value hashed from the project name in the range 6200-6455 5. The value "any", vovserver will search for an available port starting from 8000, up to 30000, until one is found. 6. The value "NUMBER:any", vovserver will first try to open the specified port, then switch to "any" if that port cannot be used.

Variable Name	Used By	Description
		<p>Examples:</p> <pre>% setenv VOV_PORT_NUMBER "automatic" % setenv VOV_PORT_NUMBER 5555 % setenv VOV_PORT_NUMBER "6000+15000" % setenv VOV_PORT_NUMBER "6273:automatic" % setenv VOV_PORT_NUMBER "any" % setenv VOV_PORT_NUMBER "7202:any" % setenv VOV_PORT_NUMBER "8690:8891:8892:8893"</pre>
VOV_PROJECT_NAME		Used by all tools. Name of the project managed by a vovserver.
VOV_PROJECT_TYPES		The primary directory where one can find the definition of project types. The default directories that are always searched are: \$VOVDIR/local/ProjectTypes and \$VOVDIR/etc/ProjectTypes .
VOV_PROTOCOL	ALL	Control the communication protocol between clients and server. The default value is "8.2". Legal values are either "8.2" or "8.3", but any value other than "8.2" is interpreted as "8.3" at this time. Introduced in 8.2.5, mostly for testing. Customers should not set this variable.
VOV_PTY_PORT_RANGE	ALL	<p>This range is to illustrate that at most one -I job can run. If a run -I job is submitted and a PTY port in this range is not available then the submission will fail to add a job.</p> <p>Example usage: setenv VOV_PTY_PORT_RANGE 14000:14000</p>
VOV_RAM_SENTRY	vovtaskerroot vovtasker	Enables the RAM Sentry, a mechanism that monitors jobs' memory utilization and prevents them from forcing a tasker to enter swap. Set this variable to 1 in the NC setup.tcl file to enable the RAM Sentry.
VOV_READONLY_PORT	vovserver	<p>Controls the port number to which vovserver will bind for guest access to the web UI. The variable can have the following forms:</p> <ol style="list-style-type: none"> 1. A number: specify a static port. 2. Two numbers separated by a + (e.g. 8000+20000) where the first number represents a base and the second number a range for the selection of a port based on a hash of the project

Variable Name	Used By	Description
		<p>name. In the example 8000+20000, the port will be selected in the range from 8000 to 28000.</p> <ol style="list-style-type: none"> 3. The value "automatic", which prompts vovserver to use the chosen web port number as a baseline, increment it by 1, and use the result as the port number. If the web port is disabled (set to 0), the numeric hash of the project/instance name added to a base number of 6200 will be used as the baseline. If the resulting port is not available, vovserver will search for an available port in increments of 1, up to the initial port + 10, until one is found. This value is accepted for first starts only, as vovserver will rewrite the variable to the appropriate configuration files in the format of N:automatic, where N is the port number that was chosen. This format provides clients with the port number on which to communicate and also the port selection mode that is to be used for subsequent starts. 4. The value "any" (default), which is identical to the automatic behavior described above, except that if the initially chosen port is unavailable, vovserver will search for an available port in increments of 1, up to 30000, until one is found. <p>Examples:</p> <pre>% setenv VOV_READONLY_PORT_NUMBER 5555 % setenv VOV_READONLY_PORT_NUMBER "6000+15000" % setenv VOV_READONLY_PORT_NUMBER "automatic" % setenv VOV_READONLY_PORT_NUMBER "6272:automatic" % setenv VOV_READONLY_PORT_NUMBER "any" % setenv VOV_READONLY_PORT_NUMBER "7201:any"</pre>
VOV_REGISTRY	vovproject vovserver	Used to point at the VOV Registry directory. By default, this is \$VOVDIR/local/registry.
VOV_RELIABLE_TIMEOUT	vovsh vovtasker	This variable controls how long the software waits for the server to appear or to reappear after a crash. The value is a "time specification". The default is 3 seconds for the first connection and 0 seconds for reconnection after a crash.

Variable Name	Used By	Description
VOV_RESOURCES	vw vw2 vov vrt	If a new job is being added to the trace by direct execution and this variable is set, it will be used as resources of the job.
VOV_RETRACE_FAST	vw vrt vtrace vw2 vov	Used to indicate to the tools that a <i>fast retracing</i> has been requested, meaning that the dependencies should not be computed. The value of the variable is irrelevant. This variable is set by <code>vw</code> . The user should not be concerned about this variable.
VOV_RETRY_CHDIR	vovtasker	Control how many times the tasker needs to try <code>chdir()</code> to the working directory of the job. Normally, the <code>chdir()</code> is quite fast. On some systems, the <code>chdir()</code> may fail for a variety of reasons, including automount failure, NFS caching delays. The default value is 1, the range is from 0 to 100, which is silently enforced.
VOV_RETRY_CHDIR_SLEEP	vovtasker	Controls the time interval between successive attempts to try <code>chdir()</code> to the working directory of the job. The default value is 1, the range is from 0 to 100, which is silently enforced.
VOV_RETRY_CHDIR_SLEEP_FACTOR	vovtasker	A factor that scales the VOV_RETRY_CHDIR_SLEEP amount between successive attempts to try <code>chdir()</code> to the working directory of the job. See also VOV_RETRY_CHDIR and VOV_RETRY_CHDIR_SLEEP. The default value is 1.0 and must be 1.0 or greater and is silently enforced.
VOV_RSH_COMMAND	vovtaskermgr	Command used in <code>vovtaskermgr</code> to start taskers.
VOV_SERVER_EXE	vovproject	Name to the vovserver binary. Defaults to <code>vovserver</code> . Normally used during testing.
VOV_SHOW_SERVICE_TIME	vovsh vovtasker vovbuild	Used only on the client side. The vovserver always sends back to the client the "cost" in milliseconds of each service performed on behalf of the client. If set, its value represent the threshold at which the expensive services are reported using <code>stdout</code> . This variable is useful to educate the user about the impact of each client call on the vovserver. If the variable is missing

Variable Name	Used By	Description
		<p>or it has a non-positive value, the showing is disabled. Otherwise the value is capped to 1000ms. Example:</p> <pre>% setenv VOV_SHOW_SERVICE_TIME 10</pre>
VOV_SH_NOCONNECT		Tell <code>vovsh</code> to not connect to the server. Use option <code>-n</code> instead.
VOV_SKIP_SERVER_LOCK	vovserver	Use if locking is disabled or not working in the filesystem containing the directory where you want to run a vovserver. To be effective, you have to set this variable to 1 before starting vovserver. Use with care. You must be absolutely sure that there is no other vovserver running in the same directory.
VOV_TASKER_LOADAVERAGE	vovtasker vovtaskerroot	When computing the load on a machine, vovtasker normally uses the largest of the 1m and 5m loads. If this variable is set to any value, then the machine load will simply be the 1m load. Implemented because of user request. Our recommendation is to not use this variable.
VOV_TASKER_NAME		When a job is running in a tasker, this variable specifies the name of the tasker.
VOV_TASKER_PROCLIMIT	vovtasker vovtaskerroot	<p>Linux platforms only</p> <p>Flags to control how to assign processes to a job. There are three flags:</p> <ul style="list-style-type: none"> • 0x1: use the process group ID • 0x2: use the session ID • 0x4: use the jobid variable found in the environment <p>As of version 2019.01u4, the default value is 0x7 (i.e. all flags are on). Previously, the default value was 0x1.</p>
VOV_TASKER_SID_DISABLE	vovtasker vovtaskerroot	<p>Linux platforms only</p> <p>Possible values are 0 and 1; default is 0. Signifies that the tasker should not create a new Linux session for itself and its child processes.</p>
VOV_SQL3_DIR		The default directory for the SQLite databases is <code>sq3</code> in the server working directory (e.g. <code>vnc.swd/sq3/.</code>) This variable allows the user to change the directory. This

Variable Name	Used By	Description
		is useful if the default directory is not in a local file-system.
VOV_STAGING_DIR	vovserver	The staging of files is allowed only from a specific directory visible by vovserver. This directory is normally PROJECT.swd/staging but this can be changed by using the variable VOV_STAGING_DIR.
VOV_STDOUT_SPEC	vw vrt vov vtrace vw2	Control the names of files used to save stdout and stderr. The value is computed by substituting the substrings @OUT@ and @UNIQUE@ and @ID@. Examples: <pre>% setenv VOV_STDOUT_SPEC .std@OUT@.@UNIQUE@ % setenv VOV_STDOUT_SPEC .std@OUT@.@ID@</pre>
VOV_STOP_SIGNALS	nc stop bkill	Specify the signals sent by nc stop. This is a comma-separated list of signals to be sent to the processes in the job. The signals include TERM INT KILL HUP USR1 USR2 CONT TSTP. In addition, EXT is supported in the format EXT:SIGNAL:INCLUDERX:EXCLUDERX:SKIPTOP (notice that we use the colon ':' instead of ',' commas). <ul style="list-style-type: none"> SIGNAL is required INCLUDERX is optional and is a regexp that matches names of processes that will be sent the signal EXCLUDERX is optional and is a regexp that matches names of processes that will NOT be sent the signal SKIPTOP is optional (default 1) and is a boolean to control whether the top process (normally vw) is skipped or not (i.e. by default we do not send the signal to vw) Example: <pre>% setenv VOV_STOP_SIGNALS TERM,INT,KILL % setenv VOV_STOP_SIGNALS EXT:TSTP:vwish % setenv VOV_STOP_SIGNALS CONT,CONT,EXT:TSTP:vwish,CONT</pre> In the case of nc stop, instead of this environment variable, one can use a property on a job with the name STOP_SIGNALS and with the same syntax. The signals are sent to the job with a delay determined

Variable Name	Used By	Description
		<p>by VOV_STOP_SIGNAL_DELAY, which by default is 3 second. A property on a job STOP_SIGNAL_DELAY can be used as well.</p> <p>If both property and env exist, the property is honored. If both NC_STOP_SIGNALS and VOV_STOP_SIGNALS are present in the environment, the value of VOV_STOP_SIGNALS will be used. If both NC_STOP_SIG_DELAY and VOV_STOP_SIGNAL_DELAY are present in the environment, the value of VOV_STOP_SIGNAL_DELAY will be used.</p>
VOV_STOP_SIGNAL_DELAY	nc stop bkill	<p>Specify the delay between signals sent by nc stop. In seconds. The default value is 3 second. If both NC_STOP_SIG_DELAY and VOV_STOP_SIGNAL_DELAY are present in the environment, the value of VOV_STOP_SIGNAL_DELAY will be used.</p>
VOV_STRICT_TRACING	vw vrt vov vtrace	<p>By default, FlowTracer does not trace the files specified by file in the exclude file (exclude.tcl) in the server working directory. If this variable is set then all files will be traced.</p>
VOV_SUSPEND_SIGNALS	vtk_job_control	<p>Specify the signals sent by vtk_job_control when action is SUSPEND</p> <p>This is a comma-separated list of signals to be sent to the processes in the job. The signals include TERM INT KILL HUP USR1 USR2 CONT TSTP.</p> <p>In addition, EXT is supported as a signal in the format EXT:SIGNAL:INCLUDERX:EXCLUDERX:SKIPTOP (notice that we use the colon ':' instead of ',' commas).</p> <ul style="list-style-type: none"> • SIGNAL is required • INCLUDERX is optional and is a regexp that matches names of processes that will be sent the signal • EXCLUDERX is optional and is a regexp that matches names of processes that will NOT be sent the signal • SKIPTOP is optional (default 1) and is a boolean to control whether the top process (normally vw) is skipped or not (i.e. by default we do not send the signal to vw)

Variable Name	Used By	Description
		<p>Example:</p> <pre>% setenv VOV_SUSPEND_SIGNALS STOP % setenv VOV_SUSPEND_SIGNALS EXT:STOP:::1</pre> <p>A property on a job with the name SUSPEND_SIGNALS can be used with the same syntax. The signals are sent to the job with a delay determined by <code>vtk_job_control -delay</code> option. If both NC_SUSPEND_SIGNALS and VOV_SUSPEND_SIGNALS are present in the environment, the value of VOV_SUSPEND_SIGNALS will be used.</p>
VOV_SWD_KEY	nc ncmgr	<p>This variable is used in the routines to find the server working directory. There are possibly many mappings for the server working directory, which you can find with <code>vovserverdir -m</code>.</p> <p>To add a new mapping, use <code>vtk_swd_set</code> in the <code>policy.tcl</code> file. The default for this variable is "unix" on UNIX systems and "windows" on Windows systems. If the value of this variable is "none", then the client will run without access to the SWD directory. This is often used with <code>VOVEQUIV_CACHE_FILE=vovcache</code></p>
VOV_SYNC_CACHE_DIR		<p>Point to a colon-separated list of cache directories in DesignSync (formerly by Synchronicity, then Enovia, now Dassault) Set to the canonical full path to the cache directory. If set, all links into the cache are NOT expanded. The recommended way to set this variable is with <code>vovequiv -p</code>. Do not use:</p> <pre>% setenv VOV_SYNC_CACHE_DIR /whatever/ location/.sync/cache</pre> <p>but rather use:</p> <pre>% setenv VOV_SYNC_CACHE_DIR `vovequiv -p / whatever/location/.sync/cache`</pre> <p>You can use VOV_DEBUG_SYMLINKS to check the behavior of this variable.</p>
VOV_UMASK	vw vov vrt	<p>Used to control <code>umask</code> inside a wrapper, it controls the permissions on the files created by the wrappers, such</p>

Variable Name	Used By	Description
		as the stdout and stderr logs. This takes any value allowed for umask. <pre>% setenv VOV_UMASK 022</pre>
<code>VOV_UNSET_VARNAMES</code>	<code>vw</code>	Specify a comma-separated list of variables to be unset in the job execution environment. This is normally used in conjunction with <code>VOV_LM_VARNAMES</code> so that license variables are cleaned before being augmented by Accelerator. This variable is honored by the VOV wrappers (<code>vw</code> , <code>vov</code> , ...) so jobs that use no wrapper (like interactive jobs) do not have access to this service.
<code>VOV_USE_COMMAS_IN</code>	<code>vovlanc</code> <code>vovlavtkncset</code>	This variable controls how the complex maps created by Allocator are defined on the Accelerator side. By default, if a resource map License:a maps to two components like License:a1 and License:a2, then LA defines the map as "License:a1 OR License:a2". If the variable <code>VOV_USE_COMMAS_IN_MAPS</code> is set to a non-zero value, then the map will be defined as "License:a1,License:a2" which is more efficient to manage in the scheduler.
<code>VOV_USE_INITGROUPS</code>	<code>vovtaskerroot</code>	Used on UNIX by <code>vovtaskerroot</code> to decide the method to use to switch a user identity and to initialize the groups. If set to a non-null value, the tasker uses <code>initgroups()</code> to initialize the groups. This call may add some load to the NIS information system. By default, <code>vovtaskerroot</code> caches the groups information for a user and uses a more economical call to <code>setgroups()</code> to initialize the groups. Use of this variable is recommended for the sites that have complicated setup of groups.
<code>VOV_USE_PS</code>	<code>vovps</code> <code>vovtaskerroot</code> <code>vovtasker</code>	Used by the routine that scans the process table. Use this variable to disable the code that scans the <code>/proc</code> filesystem and revert to a method based on running <code>ps</code> . This is only a bit slower. Use <code>vovps -a</code> to see if you need to set the variable. If the command freezes, then set <code>VOV_USE_PS</code> and try again. Also, you may want to try to use <code>truss vovps -a</code> to see which PID is causing the trouble. The same behavior is obtained with the option <code>-P</code> of <code>vovps</code> .
<code>VOV_USE_VEP</code>	<code>std.vov.aliases</code>	Used by Altair Accelerator Products setup scripts.


Variable Name	Used By	Description
	<code>std.vov.aliases.sh</code>	<p>Controls whether the 'vvp' command is active. This command sets the prompt to show the Altair Accelerator Products project and environment.</p> <p>If unset or set to a non-zero value, the vvp command will be active as an alias for csh/tcsh or a shell function for sh/ksh/bash.</p> <p>Set VOV_USE_VEP to 0 before sourcing your <code>.vovrc</code> file to disable vvp. The alias still exists, but is set to do nothing.</p>
<code>VOV_USE_VOVGROUPTASKER</code>	<code>vovtaskerroot</code>	<p>Used on UNIX-based systems by <code>vovtaskerroot</code>. By default, instead of directly calling the <code>getgrnam()</code> POSIX API function, which has been shown to hang randomly and indefinitely on some systems (e.g. Linux with LDAP), the tasker will call the external utility <code>vovgetgroups</code> to find the list of groups to which a user belongs. This variable provides two types of control over this behavior: Set to 0 to disable the use of the external <code>vovgetgroups</code> utility and force the tasker to call <code>getgrnam()</code> directly. Set to 1 to continue to use the external utility, but instruct the utility to call the <code>getgrent()</code> POSIX API function instead of the default call to <code>getgrouplist()</code>. This is mainly for debugging purposes, since this mode of operation results in slower processing of group information.</p>
<code>VOV_VW_CHECK_UPCONES</code>	<code>vw</code> <code>vov</code> <code>vrt</code> <code>vw2</code>	<p><code>setenv VOV_VW_CHECK_UPCONES 1</code></p> <p>If set to 1, tells the Vov wrappers to check the timestamp of the files in the upcone of the job. This is useful if your organization allows editing of intermediate files. This is equivalent to using the option <code>-u</code> in <code>vw</code> (or <code>vov</code>, <code>vrt</code>, <code>vw2</code>).</p>
<code>VOV_VW_ENTERPRISE</code>	<code>vw</code> <code>vov</code> <code>vrt</code> <code>vw2</code>	<p>When the environment variable <code>VOV_VW_ENTERPRISE</code> is defined, each job wrapper - <code>vov</code>, <code>vw</code>, <code>vrt</code>, <code>vw2</code> - will look for two scripts in <code>\$VOV_VW_ENTERPRISE_DIR</code> or <code>\$VOVDIR/local/ProjectTypes/ftenterprise/db/</code></p> <ul style="list-style-type: none"> • vov_job_start - executed in parallel to the job at the same time as the job starts. As it runs in the background, its output gets logged into a unique file of the form <code>.stdstart_XXX</code> similar to the <code>.stdout</code> files.

Variable Name	Used By	Description
		<ul style="list-style-type: none"> vov_job_end - executed in the background right after the job end. As it runs in the background, its output gets logged into a unique file of the form <code>.stdend_xxx</code> similar to the <code>.stdout</code> files. <p>Those scripts can be written in bourne shell, cshell, tcl or any other language.</p>
VOV_VW_ENTERPRISE	vw vov vrt vw2	<p>Used in conjunction when the environment variable VOV_VW_ENTERPRISE is defined. Use VOV_VW_ENTERPRISE_DIR to define a location where to find:</p> <ul style="list-style-type: none"> vov_job_start - executed in parallel to the job at the same time as the job starts. vov_job_end - executed in the background right after the job end. <p>If this variable is not defined, the location <code>\$VOVDIR/local/ProjectTypes/ftenterprise/db/</code> will be used instead.</p>
VOV_VW_EXE_DB	vw vov vrt vw2	<p>Choose the database (DB) of the executables used to run the command. The normal value is "FILE" and other meaningful values are "FILEX", "PHANTOM". If you specify another value, e.g. "NONE", then the executable will not be added to the input dependencies of the job. The default value is "FILE".</p>
VOV_VW_LICMON	vw vov vrt vw2	<p>If this variable is set, then the wrappers react to VOV_LICMON by calling <code>ftlm_lmproject</code>. If the variable is not set, then only the vovtasker reacts to VOV_LICMON.</p>
VOV_VW_LIVERECORDER	vw vov vrt vw2	<p>Specify directory for LiveRecorder log file for the job wrapper. Default is <code>/tmp</code>.</p>
VOV_VW_LIVERECORDER_SIZE	vw vov vrt	<p>Specify maximum size (MB) of LiveRecorder log file for the job wrapper. Default is 256.</p>

Variable Name	Used By	Description
	vw2	
VOV_VW_LIVERECORDER	vw vov vrt vw2	Set to any value to enable LiveRecorder for the job wrapper.
VOV_VW_LOGNAME	vw vov vrt vw2	Choose the name of the logfile used for verbose output. The default has the form <code>.vw2_PID_verbose.out</code>
VOV_VW_NFS_PROTECTION	vw vov vrt vw2	<code>setenv VOV_VW_NFS_PROTECTION 1</code> This is equivalent to the option <code>-n</code> in <code>vw</code> . If set to 1, tell vov wrappers to activate the protection (retry up to 60 seconds at 5 seconds intervals) against dirty NFS caches. You can find that a particular job has needed that protection by looking for a <code>NFS_PROTECTION</code> property attached to the job.
VOV_VW_NFS_WAIT	vw vov vrt vw2	This variable controls the wait time for the NFS protection mechanism. The acceptable range is between 10s and 1h. The default value is 1m. Example: <pre>setenv VOV_VW_NFS_WAIT 10m</pre>
VOV_VW_PAUSE_ON_TSTP	vw	Determines whether to pause <code>vw</code> jobs when they receive a SIGTSTP signal. Possible values are 0 and 1; default is 0. If set to 1, <code>vw</code> jobs will be paused when they receive a SIGTSTP signal and continue when receiving either a SIGCONT or SIGALRM signal. Example: <pre>% setenv VOV_VW_PAUSE_ON_TSTP 1</pre>
VOV_VW_PING	vw vov vrt	For sites that have nasty firewalls that close inactive connections, you can use this variable to force <code>vw</code> to keep the connection alive. For example, <pre>setenv VOV_VW_PING 20m</pre>

Variable Name	Used By	Description
	vw2	will send a message ('ping') every 20m The default is 20m. The value of VOV_VW_PING is silently restricted between 3s and 10d. This function is disabled by setting VOV_VW_PING to 0. A good place to set this variable is the <code>setup.tcl</code> file.
VOV_VW_RETRY_WHICH	vw	Control how many times the wrapper will try to verify the existence of a job's command. This is mainly useful when submission scripts are used that generate a job script, and that script is located on an NFS file share that often experiences delays. The default value is 0 (try once, no retries), the range is from 0 to 100, which is silently enforced.
VOV_VW_RETRY_WHICH_SLEEP	vw	Controls the time interval (in seconds) between successive attempts to verify the existence a job's command. The default value is 1, the range is from 1 to 100, which is silently enforced.
VOV_VW_RETRY_WHICH_SLEEP_FACTOR	vw	A factor that scales the VOV_VW_RETRY_WHICH_SLEEP amount between successive attempts to try verify a job's command. The default value is 1.0, the range is from 1.0 to 10.0, which is silently enforced.
VOV_VW_SIG_IGN	vw vov vrt vw2	Tell vov wrappers to ignore most signals. Same behavior as option -s.
VOV_VW_TRACK_LINKS	vw vov vrt vw2	Tell vov wrappers to track all symbolic links used in path expansion.
VOV_VW_VERBOSE	vw vov vrt vw2	Make vov wrappers verbose. Same as -v. Use values from 0 to 4.
VOV_WEB_PORT_NUMBER	vovserver	Controls the port number to which vovserver will bind for guest access to the web UI. The variable can have the following forms:

Variable Name	Used By	Description
		<ol style="list-style-type: none"> 1. A number: specify a static port. 2. Two numbers separated by a + (e.g. 8000+20000) where the first number represents a base and the second number a range for the selection of a port based on a hash of the project name. In the example 8000+20000, the port will be selected in the range from 8000 to 28000. 3. The value "automatic", which prompts vovserver to create a numeric hash of the project/instance name, add it to a base number of 6200, and use the result as the web port number. If the resulting port is not available, vovserver will search for an available port in increments of 1, up to the initial port + 10, until one is found. This value is accepted for first starts only, as vovserver will rewrite the variable to the appropriate configuration files in the format of N:automatic, where N is the port number that was chosen. This format provides clients with the port number on which to communicate and also the port selection mode that is to be used for subsequent starts. This port will be used to seed the selection process for the VOV_READONLY_PORT_NUMBER. 4. The value "any" (default), which is identical to the automatic behavior described above, except that if the initially chosen port is unavailable, vovserver will search for an available port in increments of 1, up to 30000, until one is found. <p>Examples:</p> <pre>% setenv VOV_WEB_PORT_NUMBER 5555 % setenv VOV_WEB_PORT_NUMBER "6000+15000" % setenv VOV_WEB_PORT_NUMBER "automatic" % setenv VOV_WEB_PORT_NUMBER "6271:automatic" % setenv VOV_WEB_PORT_NUMBER "any" % setenv VOV_WEB_PORT_NUMBER "7200:any"</pre>
VOV_WX_STUCK_VOVMX	live_wx_check_vovwxd	<p>Age at which the vovwxd daemon should be considered as stuck and an alert should be generated.</p> <p>Set in wx.swd/setup.tcl. Age can be in timespec or seconds format.</p>
VOV_WX_STUCK_VOVMX	live_wx_check_vovwxd	<p>If set to 1, the vovwxd daemon will be restarted if the stuck age has been exceeded.</p>

Variable Name	Used By	Description
		Set in <code>wx.swd/setup.tcl</code> .
<code>VOV_WX_STUCK_VOVWX</code>	<code>live_wx_check_vovwxd</code>	<p>If set to 1, the <code>vovwxd</code> daemon will be straced if the stuck age has been exceeded.</p> <p>Set in <code>wx.swd/setup.tcl</code>.</p> <div>  Note: Both the timeout and strace commands must be on the system path for this functionality to be enabled. </div>
<code>WX_NOCLEAN_CMDFILE</code>	<code>wxrun</code>	<p>Performance optimization for Accelerator Plus job submissions.</p> <p>Setting to 1 will disable line cleaning in <code>wxrun</code> when using a command file (-f). Doing so puts the burden of making sure the command file has clean line endings and appropriately escaped characters, when needed.</p>
<code>WX_PREJOB_PER_TASK</code>	<code>bsub</code>	<p>Performance optimization for Accelerator Plus job submissions using the LSF <code>bsub</code> emulation utility.</p> <p>Setting to 1 will change the behavior of specifying a pre-job in such a way that it is done on a per-task basis instead of a per-job basis. This is beneficial when the pre-job is used to validate that a host is appropriate for a job and no per-job functionality is being performed.</p>

Other Files Used by VOV

Table 1:

File	Description
<code>\$HOME/.vovrc</code> or <code>\$HOME/.vovrc.sh</code>	This file is created by <code>vovsetupuser</code> . It is updated by <code>vovproject create</code> to store aliases related to various projects. It is normally sourced automatically by the shell.
<code>\$VOVDIR/local/license.key</code>	This is the default license file.
<code>\$VOVDIR/etc/std.vov.aliases</code>	The Standard VOV Aliases for the C-shell. This file is normally sourced from your <code>.vovrc</code> file.

File	Description
<code>\$VOVDIR/etc/ std.vov.aliases.sh</code>	The Standard VOV Aliases for the Bourne shell. This file is normally sourced from your <code>.vovrc.sh</code> file.

HTML Pages

The vovserver can also locate HTML pages in the following locations:

User's Home Directory

The URL has the form: `"/~<user>/<filespec>"` and the file is found in a subdirectory of `"~user/public_html"`

HTML root directory

The URL has the form: `/html/<filespec>` and the file is found according to the following search order:

1. `$VOV_PROJECT_DIR.swd/html/` (relative to the server working directory)
2. `html/` (relative to the server working directory)
3. `$VOVDIR/local/html/`
4. `$VOVDIR/etc/html/`

If the requested file cannot be found, a simple error page is returned.

Raw HTTP/VOV Interface

/raw/label/id

Return a label for node with the given ID. This interface is used by the applets that display the grid and graph.

Examples:

```
/raw/label/000290903
```

/raw/jobqueue

Returns the number of jobs in the queue.

/raw/microevents

This URL is associated with a **stream of events** where each event has the following format:

```
subject verb id aux
```

where

subject

Is an integer defined by `vovinternal.h`. In particular, 0=node 1=place 2=transition 3=annotation 5=nodeset

verb

Is an integer with the following meaning defined by `event.hh`. In particular, 1=create 2=forget 6=connect 7=disconnect 8=change 14=overflow

id

Is the ID of the entity involved in the event

aux

Is either 1 or 2 integers specifying more information about the event.

- If the verb is CHANGE, then *aux* is an integer representing the new node status.
- If verb is MOVE, then *aux* is two integer specifying the new X and Y location of the node
- Otherwise, *aux* is the VovId of another node involved in the event

/raw/grid

This URL accepts the following options:

Option	Description
id	ID of the set
name	Name of the set. If both name and id are specified, name prevails.

Examples:

```
/raw/grid?id=0034345
```

The URL returns list of **nodes** in the specified set using the following format:

```
id type x y status  
label
```

where:

id

Is the VovId of the node

type

Is an integer: 1=file, 2=job, 5=barrier file

x

The x coordinate of the node

y

The y coordinate of the node

status

Is an integer expressing the status of the node (0=valid, 1=invalid, 3=running, ...)

label

On a line by itself, is a string of characters to be associated with the node.

The list is terminated by an empty line.

/raw/graph

This URL returns a list of **nodes** followed by a list of **arcs**. The options are the same as for /raw/grid. The format of the list of nodes is also the same as for /raw/grid. Each arc is expressed by a pair of VovId's:

```
fromId toId
```

and is terminated by an empty line.

/raw/taskers

This URL returns the encoded information about the status of the taskers.

```
id name status statuscolor
```

where:

id

Is an integer which is the tasker identifier

name

Is a string which is the name of the tasker

status

Is a string that represents the status of the tasker (e.g. Idle, Working, ...)

statuscolor

Is a color associated with the tasker status.

The list is terminated by an empty line.

/raw/resource

Show the utilization of the specified resource over time. The following options are accepted:

Option	Description
name	The name of the resource. This option is required.
start	The start of the time window (UT). The default is the time corresponding to one week ago.
finish	The start of the time window (UT). The default is the start time plus one week.

If the finish time is before the start time, the URL silently returns an empty file.

Examples:

```
/raw/resource?name=diskio
```

```
/raw/resource?name=diskio&start=950138888&finish=950200000
```

The output consists of pairs of the form **(time quantity)** where time is either absolute (if it begins with a digit) or relative to the previous one if it begins with a + sign. Example:

```
949544831 1
+10 0
+5 1
+4 0
+3 1
+3 0
+3 1
+3 0
+2 1
+2 0
+2 1
+52 0
+2 1
+3 0
950146174 0
```

/raw/tool

Same as /raw/resource.

VOV Security Infrastructure

VOV Security Keys

The VOV system supports a public/secret key pair-based security and authentication mechanism called *VOV Security Keys*. VOV security key authentication is the preferred authentication mechanism for REST and is also the basis for highly secure connections between Accelerator and the Event port on Monitor when RDS resource management is activated.

VOV security keys are based on ed25519 public key cryptography. Conceptually, they are like SSH keys:

- You create a public/private key pair.
- You use some other authentication means to connect to the server, and tell the server to trust your public key and associate it with your identity.
- You use your private key as part of an authentication "handshake" with the server, and it verifies your identity using your matching public key.

It's important to remember that you do not need a new public/private key pair for each queue where you intend to use key based authentication. You can create one key pair and set up the public half of your key on all the vovservers you wish to authenticate against.

VOV security keys in no way replace the security rules and policies enforced by `security.tcl` or Access Control Lists. The keys are only a means to authenticate the user, after which Security Levels and Access Control Lists still come into play.

Create a Public/Private Key Pair

Method 1 - Using the VOV CLI

If the host where the REST client will execute has access to the VOV software installation, you can use these VOV CLI commands:

```
% vovproject enable PROJECT
% vovsecurity keygen
```

By default, `vovsecurity keygen` writes the newly generated key pair into `$HOME/.vov/userkey`. If that file already exists, you will be prompted on whether you wish to overwrite the file. Overwriting the file means you will no longer have access to the previous key pair that you may have used to establish key based authentication on other vovserver instances.

Method 2 - Using the Browser

If the client host does not have access to the VOV command line interface on the same local network where the vovserver is running, then this method will work.

Browse to URL `<VOV_URL>/api/v3/apikeys/newkey`

The browser will display a generated random key data in a JSON format. The displayed data shows the new secret key followed by the corresponding public key.

Method 3 - Using Curl

This is equivalent to the prior method, but using the Linux curl command to access the target URL:

```
% curl -k -X GET <VOV_URL>/api/v3/apikeys/newkey | jq '.rows[0]'
```

Register Your Public Key on a vovserver Instance

Method 1 - Using the VOV CLI

If you have access to a host on the LAN where the vovserver is running and the VOV CLI commands, the CLI key registration method is convenient. Normally the public VOV security key is stored in `$HOME/.vov/userkey`, and the `vovsecurity getkey` command simply reads the public key. The `vovsecurity addkey` command will prompt for this user's password to authenticate the registration request, which is implemented behind the scenes as an authenticated REST request.

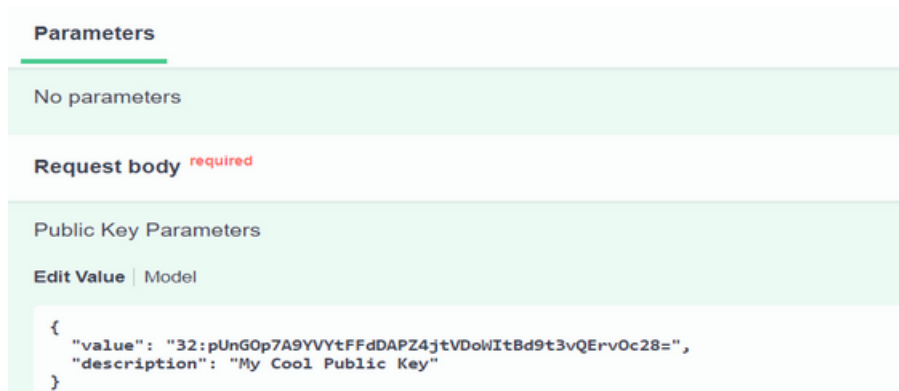
```
% vovproject enable PROJECT
% pubkey=$(vovsecurity getkey)
% vovsecurity addkey -kv $pubkey -kd "My Cool Key"
Enter your password for connecting to PROJECT on HOST: *****
```

To script the above sequence without being interactively prompted for a password, set the `VOV_PASSWORD` environment variable to the value of your password before invoking the `vovsecurity addkey` command.

Method 2 - Using the VOV REST API Interactive Web UI Page

The VOV project's web UI provides a means to register a user's public key at vovserver. To register a public key, follow this procedure:

1. Browse to the VOV REST API Interactive Web UI Page at `<VOV_URL>/html/vovrest.html`
2. Login with the user name that corresponds to the public key being registered.
3. Scroll down to the **apikeys** topic and click on the **POST** button beside the **/apikeys Add Key** line.
4. Click **Try it Out** on the right side of the page.
5. Edit the Request Body JSON to add your public key for the "value" keyword, and to add a description string for the "description" keyword.
6. Click **Execute**.



Parameters

No parameters

Request body required

Public Key Parameters

Edit Value | Model

```
{
  "value": "32:pUnG0p7A9YVYtFFdDAPZ4jtVD0WItBd9t3vQErv0c28=",
  "description": "My Cool Public Key"
}
```

Figure 1:

Listing Your Public Keys on a vovserver Instance

Method 1 - Using the VOV CLI

You can list the public keys that have already been registered on a vovserver instance, using the following command in an enabled shell:

```
% vovsecurity listkeys
user1 32:Ihq6djlX5L9XQzRWRWy0Z2h15fD64JhJoqa7FpDmiyg= Backup key
user1 32:+fmR1SmWValRspQjcLQ7OGX266MZj/m90B5fii3FMTY= 2nd key
user1 32:wlefJq4QjTm2QgtWTUtD7kG1lnCyUtrKfobW/HPfoD0= 3rd Key
```

In this case, user1 has registered multiple public keys on the same vovserver instance.

Method 2 - Using the VOV REST API Interactive Web UI Page

The VOV project's web UI provides a means to list a user's registered public keys. Follow these steps:

1. Browse to the VOV REST API Interactive Web UI Page at <VOV_URL>/html/vovrest.html
2. Login with the user name that corresponds to the public key being registered.
3. Scroll down to the **apikeys** topic and click on the **GET** button beside the **/apikeys List Key** line.
4. Click **Try it Out** on the right side of the page.
5. Click **Execute**.
6. Scroll down to the Response Body sub-window and view the JSON description of the list of registered public keys.

Using Key Based Authentication from a Python-Based REST Client

REST client authentication is typically done through VOV security keys that are generated and properly registered with a vovserver instance and stored in a local file on the client. The key based authentication is handled automatically inside the submitRequest() Python REST member function described below.

To perform key based authentication with the REST API, the following must be true:

- The vovserver webport must be non-zero (this is true by default).
- The vovserver webprovider must be set to "internal" (this is the default value).

- The `ssl.enable configuration` parameter should be set to 1 (this is the default value)
- A VOV security key pair should be generated and stored on the client host in a file (normally `~/.vov/userkey`), in the following format. This can be done using the `vovsecurity keygen` command or the REST program shown in [Create a Public/Private Key Pair](#) above.

```
% cat ~/.vov/userkey
secret-key = 32:WUhyiwOUBYqonfVR66fNVnHd6sfK9QbK257A8jheYfc=
public-key = 32:NIqRzIOhv0r7GZFTPkL0nQLZX6U6UTihleHhFwlVqnQ=
```

- Key pairs must be registered with a vovserver via the `vovsecurity addkey` command or the REST program shown in [Register Your Public Key on a vovserver Instance](#) above.

The simplest way to program a REST request using key-based authentication is to pass the two required arguments to the `submitRequest()` method function for the `VOVRestV3` class in the `vov_rest_v3.py` module that is included in `$VOVDIR`. Before running this Python program, perform the following setup from an enabled shell:

```
% export VOV_URL=$(vovbrowser)
% cp $VOVDIR/scripts/python/vov_rest_v3.py .
```

This Python program can be run with Python version 3 or higher. The `submitRequest()` function will automatically take care of these prerequisite actions before issuing a REST request:

1. Retrieve the server public key
2. If the first REST call, or if the underlying REST access token is expired, then the user will be authenticated using the secret VOV key from `~/.vov/userkey`. If the secret VOV key is stored in a different file, then add the keyword parameter `authKeyFile=filename` to the argument list in the `submitRequest()` function call.

```
#
# easy_REST_key_101.py
#
import os, sys, vov_rest_v3
url = os.environ['VOV_URL']
vrest = vov_rest_v3.VOVRestV3()
r = vrest.submitRequest("GET", url + "/api/v3/project/1/project")
print (r)
```

The following example shows an alternative to the above. It explicitly authenticates using the `authorizeWithKey()` method function prior to calling `submitRequest()`. This allows one user impersonate another user whose secret key is known. It also allows explicit passing of the secret key as an argument instead of passing the name of a local file containing the key.

Before running the Python program below, take these preparatory steps:

- Set `VOV_URL` and copy `vov_rest_v3.py` locally:

```
% export VOV_URL=$(vovbrowser)
% cp $VOVDIR/scripts/python/vov_rest_v3.py .
```

- Set the `MY_SECRET_KEY` environment variable to the user secret VOV security key (normally viewed in `~/.vov/userkey`)

- Set the VOV_SERVER_PUBLIC_KEY environment variable to the value of the server public key that is assigned to a VOV project at creation time. There are three ways to do this:

```
# Method 1
Browse to URL    <VOV_URL>/api/v3/apikeys/serverkey

# Method 2
% curl -k -X GET    <VOV_URL>/api/v3/apikeys/serverkey | jq '.rows[0][0]'

# Method 3
% vovproject enable PROJECT
% vovsecurity getserverkey
```

This example explicitly calls the `authorizeWithKey()` method function to authenticate, followed by the `submitRequest()` function to issue the REST request. This might be convenient if multiple key pairs are registered with the vovserver for this user, and the desired key pair is not currently present in `~/.vov/userkey`.

```
#
# explicit_REST_key_101.py
#
import os, sys
import vov_rest_v3

url = os.environ['VOV_URL']
secret_key = os.environ['MY_SECRET_KEY']
sp_key = os.environ['VOV_SERVER_PUBLIC_KEY']
user = os.environ['USER']
vrest = vov_rest_v3.VOVRestV3()
vrest.authorizeWithKey(url, secret_key, sp_key, username=user)
r = vrest.submitRequest("GET", url + "/api/v3/project/1/project")
print (r)
```

Advanced Key Handling

The REST client will use the libsodium library to construct an encryption message which is a combination of the user's private key and vovserver's public key, which will be sent to REST at the endpoint `/api/v3/token` with `grant_type` set to `apikey` and `apikey` set to the contents of the encrypted message.

If you wish to handle REST sessions yourself, look inside `vov_rest_v3.py` at the `_getEncryptedMessage()` procedure to see how the authentication handshake message is composed using the user's private key and the vovserver's public key. In `_getEncryptedMessage()`, you can see how it decodes the two key strings into raw bytes, creates a PyNaCl Box object using the two keys, and then encrypts the message by calling `Box.encrypt()`. The encrypted message is then base-64 encoded and returned as a string that the vovserver REST API can understand.

```
def _getEncryptedMessage(self, userseckey, serverpubkey, message=None):
    """Generated an encrypted message block using the caller's secret key and the
    server's public key.

    :type userseckey string
    :param userseckey User's secret key. It should be in base64 format with the
    binary length prepended,
                        and a colon separating the length and the base64 string.

    :type serverpubkey string
```

```
:param serverpubkey vovserver's public key. It should be in base64 format with
the binary length prepended,
    and a colon separating the length and the base64 string.
    This can be retrieved from $VOVDIR/local/registry/ as the
'CLIENT_PUBKEY' entry.

:type message string
:param message The message to encode. Optional. If None, a generic default
message is used. It doesn't
    effect the key authentication process in any way.

:rtype: string
:return: A string containing a base64 encoded, then url-encoded buffer.
"""
idx = userseckey.find(':')
if idx == -1:
    return ""
buflen = int(userseckey[:idx])
userseckey = userseckey[idx + 1:]
seckeybuf = base64.b64decode(userseckey)
seckeybuf = seckeybuf[:buflen]

idx = serverpubkey.find(':')
if idx == -1:
    return ""
buflen = int(serverpubkey[:idx])
serverpubkey = serverpubkey[idx + 1:]
pubkeybuf = base64.b64decode(serverpubkey)
pubkeybuf = pubkeybuf[:buflen]

if message is None:
    message = 'VOV API Key Client Auth ' + str(time.time())
messageBuf = message.encode("utf-8")
secretKey = PrivateKey(seckeybuf)
publicKey = PublicKey(pubkeybuf)

box = Box(secretKey, publicKey)
cipher = box.encrypt(plaintext=messageBuf)
cipherLen = len(cipher)
encodedCipher = base64.b64encode(cipher)
cipherStr = str(cipherLen) + ':' + encodedCipher.decode("utf-8")
return cipherStr
```

The following code snippet demonstrates how to get a REST session token using the `_getEncryptedMessage()` procedure above:

```
url = 'https://hostname:9100/api/v3/token'
clientSecret = ''
postData = {}
postData["grant_type"] = "apikey"
postData["client_id"] = "Python client"
postData["client_secret"] = clientSecret
postData["username"] = "myusername"
postData["apikey"] = self._getEncryptedMessage(userseckey, serverpubkey)
session = requests.Session()
response_obj = session.post(url, timeout=self.connectionTimeout, verify=False,

                            data=postData)
respContent = response_obj.text
respStatus = response_obj.status_code

if (respStatus == 200):
```

```
contentJSON = json.loads(respContent)
if ("access_token" in contentJSON):
    sToken = contentJSON['access_token']
```

Once you have the session token, you can use it for subsequent REST API requests, similar to previous REST examples which used username/password authentication to get the session token.

Deleting A Key from vovserver

You can delete your own key from vovserver using the `vovsecurity` command:

```
vovsecurity delkey -kv 32:jA0iq1mLbdSuuDKj1PeEoHJX+fYqR/0HxpI5nhgThE8=
```

Users with ADMIN level security access in the project can list and delete keys belonging to other users:

```
> vovsecurity listkeys -a
user1          32:jA0iq1mLbdSuuDKj1PeEoHJX+fYqR/0HxpI5nhgThE8= Primary Key
user2          32:Ihq6djlX5L9XQzRWRWy0Z2h15fD64JhJoqa7FpDmiyg= Primary Key
user2          32:+fmR1SmWValRspQjcLQ7OGX266MZj/m90B5fii3FMTY= 2nd key
user2          32:wlefJq4QjTm2QgtWTUtD7kG11nCyUtrKfobW/HPfoD0= 3rd Key
> vovsecurity delkey -kv 32:jA0iq1mLbdSuuDKj1PeEoHJX+fYqR/0HxpI5nhgThE8= -u user1
```

vovsecurity

This program provides security key pair management functions for VOV.

```
vovsecurity : Usage Message

vovsecurity ACTION {OPTIONS}

vovsecurity - security and key related management commands:
This program provides security key management functions for VOV.

ACTIONS:
  keygen          Generate a new VOV security key pair and store them in
                  ~/.vov/userkey
  getkey          Read your VOV public key from ~/.vov/userkey and print to
                  stdout
  getserverkey    Show the VOV server public key
  addkey          Add a user's public key to vovserver
  delkey          Delete user key from vovserver
  listkeys        List user's public keys currently registered on vovserver
  gettoken        Get a token for a given security level and duration
OPTIONS:
  -f <filename>   Filename to read or write for API keys.
                  The default is ~/.vov/userkey
  -kv <keyval>    Value of public key to add or delete
  -kd <keydes>    Human readable description of key being added
  -sl <securitylevel>
                  Security level for token
  -du <duration>  Duration for token
  -y              Don't prompt when overwriting existing file
  -v              Enable verbosity
  -a              List public keys for all users. Only valid for admins
```

```
-stdout      Print key to stdout
-u <username> Specify user whose key you wish to delete. Only valid for
              admins
-h           Show brief usage
```

EXAMPLES:

```
% vovsecurity keygen
% vovsecurity keygen -f keyfile.txt -y
% vovsecurity getkey
% vovsecurity getserverkey
% vovsecurity addkey
% vovsecurity addkey -kv KEYVAL -kd 'My New Key'
% vovsecurity delkey -kv KEYVAL
% vovsecurity delkey -kv KEYVAL -u USERID
% vovsecurity listkeys
% vovsecurity listkeys -a
% vovsecurity listkeys -u USERID
% vovsecurity gettoken -sl SECURITYLEVEL -du DURATION
```

Examples

Action	Command
Generates a public/private key pair and tries to write to <code>\$HOME/.vov/userkey</code> . If the file already exists, you will be asked whether to overwrite the existing file. Overwriting your key file means that the previous keypair stored there will no longer be usable, and thus rendering storage of that public key on any vovserver instances as useless.	<pre>> vovsecurity keygen</pre>
Generates a public/private keypair and writes it into a separate key file named <code>keyfile.txt</code> .	<pre>> vovsecurity keygen -f keyfile.txt</pre>
Reads <code>\$HOME/.vov/userkey</code> and echos the public key back to the console.	<pre>> vovsecurity getkey</pre>
Retrieves the public key for the current project. The vovserver's public key is written in <code>\$VOVDIR/local/registry/[system-nc system-wx system-lm]</code> folders respectively. Currently, FlowTracer and Allocator projects do not support the system registry.	<pre>> vovsecurity getserverkey</pre>
Adds a new public key to vovserver, associated with the current user. Interactively prompts for the key value and the key description on the terminal,	<pre>> vovsecurity addkey</pre>

Action	Command
in addition to the user's password in order to perform password based authentication.	
Adds a new public key to vovserver associated with the current user. Interactively prompts for the user's password in order to perform username/password authentication with vovserver. If VOV_PASSWORD is set, the value of this environment variable will be used, and no prompts will be made.	<pre>> vovsecurity addkey -kv KEYVAL -kd 'My New Key'</pre>
Deletes a key belonging to the current user. Interactively prompts for the user's password in order to perform username/password authentication with vovserver. If VOV_PASSWORD is set, the value of this environment variable will be used, and no prompts will be made.	<pre>> vovsecurity delkey -kv KEYVAL</pre>
For users with ADMIN security access on the project, keys belonging to other users can be deleted using this variation.	<pre>> vovsecurity delkey -kv KEYVAL -u USERID</pre>
Lists all the current user's public keys which vovserver has stored. Interactively prompts for the user's password in order to perform username/password authentication with vovserver. If VOV_PASSWORD is set, the value of this environment variable will be used, and no prompts will be made.	<pre>> vovsecurity listkeys</pre>
For users with ADMIN access to the project, list all public keys stored in vovserver for the project. Interactively prompts for the user's password in order to perform username/password authentication with vovserver. If VOV_PASSWORD is set, the value of this environment variable will be used, and no prompts will be made.	<pre>> vovsecurity listkeys -a</pre>

Client User Authentication

Client user authentication for clients connecting to the VOV port on vovserver is implemented "opt-in" in this release. To enable this authentication, set the `comm.requireClientAuth` server configuration

parameter to 1. Without a valid key registered in the server, you will have a limited set of commands you can perform against the queue without error. These include `vovproject enable`, `vovbrowser`, `vovsecurity`, and `nc cmd vovsecurity`.

When client user authentication is enabled, VOV client and server will authenticate the user using the VOV security key pair stored in `~/.vov/userkey` if present (`%USERPROFILE%\vov\userkey` for Windows clients). These are the same security keys used to authenticate REST clients, and are managed via the `vovsecurity` command. This affects system clients like taskers and VOV daemons as well as end user clients like job submit and job status check commands.

There are administrative requirements for the VOV project-owning user as well as any user who will run jobs on an Accelerator, Accelerator Plus, FlowTracer, or Hero products before client authentication should be enabled. The following prerequisites must be taken care of before running a VOV project with client authentication enabled:

1. The project owner must have created a key pair using `vovsecurity keygen` that is stored at `$HOME/.vov/userkey`
2. If the VOV project is newly created, it will automatically have the owning user's public key added because project creation reads the key from `$HOME/.vov/userkey`. But if the VOV project is pre-existing, the owning user needs to add the public key to vovserver using the `vovsecurity addkey` command.
3. For Accelerator/Accelerator Plus/FlowTracer/Hero products, all users who will run jobs and query job status must create a key pair using `vovsecurity keygen` and add the public key to vovserver using the `vovsecurity addkey` command.



Note:

- Root user does not authenticate. `vovtaskerroot` taskers will "su" to the project-owning user's UID and similarly authenticate with the project owner's key.
- Running taskers or other commands in an environment without access to a key pair in `$HOME/.vov/userkey` is not supported.
- The command `vovproject enable [PROJECTNAME]` will set an additional new environment variable called `VOV_SERVER_PUBKEY` to the value of the server instance's public key. This is needed internally by clients as part of the key-based authentication process.

REST Application Programming Interface

Altair Accelerator supports a RESTful API that enables a developer to submit, monitor, and control batch jobs as well as query the batch scheduler queues, jobs, and hosts in useful ways.

Overview

Altair Accelerator supports a Representational State Transfer (REST) API that enables a developer to submit, monitor, and control batch jobs as well as query the batch scheduler queues, jobs, and hosts in useful ways. A REST API is based on a URL name space targeted by HTTP operations like POST and GET to submit jobs or make queries. REST APIs are popular ways for developers to write portable applications, both UI and command line, that can interface to the Accelerator queue over secure HTTPS connections. This tutorial will introduce key concepts and will also provide working example Python programs that interface to Accelerator via REST.

Developers use a variety of languages in web clients when interfacing to a REST API, including Python, PHP, Java, Node.js and others. This tutorial will focus on the use of Python in example code.

Key REST Concepts

The components of a REST operation are the following:

- Resource Path
- HTTP Verb
- Body
- Header

The URLs in the application's REST URL name space provide the targets for the read (GET), write (POST), or update (PATCH) HTTP request types. The parameters of the REST operations always include a URL and an access token that authorizes access to the services. The following sections describe more about these concepts.

Resource Path

The resource path describes the object to be acted upon. It is in the form of a URL with the following structure:

```
BASE_URL + "/api/" + VERSION + "/" + REST
```

`BASE_URL` is the initial part of the URL for the Accelerator queue to access. For example, if the Accelerator `NC_QUEUE` environment variable was set to `My_Test_Queue`, then use the `nc cmd vovbrowser` command to obtain a `BASE_URL`.

The `VERSION` is the REST API version, currently v3.

REST is the remaining part of the URL that applies to the specific types of objects you will be referencing.

The resource path might look something like this when you are referencing the version information about the Accelerator queue scheduling server known as vovserver.

```
https://server.domain.myco.com:6330/api/v3/project/1/version
```

HTTP Verb

The verb describes the action to take regarding the resource.

POST	Creates a resource
GET	Retrieves one or more resources
PUT	Updates or controls a resource
PATCH	Updates a resource
DELETE	Deletes a resource

Body

The body of a REST request defines parameters and options that apply to the action being taken. A POST or PUT request has a body. A GET or DELETE request has no body. A Python dictionary data structure is the usual format for the body, and it consists of a set of keywords with associated values.

Header

The header contains metadata about the message being sent to the server which includes, importantly, an *access token*. REST HTTP requests are initiated across the network from a node other than the one on which the Accelerator server is running. The access token provides the assurance that the sender has permission to request the REST action on the server.

An authentication request type is available via a POST that supplies a username and password to the server, to which the server responds with the access token. When a REST request comes in later with this access token, the server can quickly and reliably extract the user identity and permissions as a prelude to processing the request. Accelerator REST utilizes a style of access token known as JSON Web Tokens (JWTs).

Resource Path URLs

Request URLs can be in any of the following forms. These examples assume vovserver is listening on port 2345 of a machine named myserver.

Form 1: Querying for All Fields of an Object

```
http://myserver:2345/api/v3/jobs/12345  
http://myserver:2345/api/v3/jobs/12345/
```

```
http://myserver:2345/api/v3/jobs/12345/*
```

The v3 part of the API refers to the current API revision. Old API revisions will continue to be supported as new revisions are introduced, e.g. when the current revision is 4, the v3 URLs will remain functional.

All of the above URL formats are identical in behavior and output, and will return a formatted list of all fields in job 12345. This is equivalent to `vovselect '*' from jobs where idint==12345` in the CLI.

Form 2: Querying for a Specific Field

```
http://myserver:2345/api/v3/jobs/12345/status
```

This form will return a single field (status) from job 12345. This is equivalent to `vovselect status from jobs where idint==12345`. The output will be similar to:

```
{ "startrow":1, "endrow":1, "query":"SELECT status FROM jobs WHERE  
  idint==1158062", "errmsg":"","columns":[{"col":0, "field":"status"}], "rows":  
  [{"VALID"}]}
```

Form 3: Perform a Complex Query

```
http://myserver:2345/api/v3/query?select=idint,host,status&from=jobs&where=status%3d  
%3dVALID&limit=10&order=host%20DESC
```

This form allows complex SQL-style queries (note that characters such as space and "=" must be URL encoded). The where clause contains [Selection Rules](#), which filters the data set down to one or more specific objects. This above example is equivalent to `vovselect idint,host,status -from jobs -where status==VALID -limit 10 -order "host DESC"`. The output will be similar to:

```
{ "startrow":1, "endrow":8, "query":"SELECT idint,host,status FROM jobs  
WHERE status==VALID ORDERBY host DESC LIMIT 10", "errmsg":"","columns":  
  [{"col":0, "field":"idint"},  
    {"col":1, "field":"host"}, {"col":2, "field":"status"}], "rows":  
  [[1158051, "titanus", "VALID"] [1158058,  
    "titanus", "VALID"] [1158049, "skull", "VALID"] [1158062, "skull", "VALID"]  
    [1158109, "skull", "VALID"] [1158117,  
    "skull", "VALID"] [1158101, "localhost", "VALID"] [1158105, "localhost", "VALID"]]]}
```

All forms allow the use of an additional format parameter in the URL to specify a format for the returned data. If no format is specified, the data are returned in JSON format. The following formats are currently supported:

Table 2:

Format	Description
json (default)	Return a JSON-formatted list of fields
csv	Return fields formatted as a CSV, with column headers
csv-noheader	Return fields formatted as a CSV with no column headers

For example the fields of job 12345 can be returned as a CSV list using the URL:

```
http://myserver:2345/api/v3/jobs/12345?format=csv
```

A complete list of fields, their data types, and the objects to which they belong can be viewed in [Node Fields](#).

Get Ready for REST

REST API usage prerequisites are described in this section. These prerequisites will enable REST v3 applications to run with Accelerator. Some of the prerequisites apply to the Accelerator queue configuration and others apply to the host where the REST application will run, also called the "REST client".

Prerequisites for the Accelerator queue:

- The web port must be configured. In version 2021.2.0 through 2022.1.1, the web provider must be set to "internal". See the `-webport` and `-webprovider` options in the `-h` help screen documentation for VOV project start commands like `ncmgr start` and `lmmgr start`.
- **Optional:** SSL/TLS should be enabled for security reasons. Since passwords are passed with HTTP authorization requests to vovserver, the security of the connection is important. Enable SSL/TLS for the NC queue as follows:

1. Add this line to `SWD/policy.tcl`:

```
set config(ssl.enable) 1
```

2. If REST requests will be sent from Python scripts running on CentOS 7 or earlier, TLS 1.2 will be used by the REST application you are writing. Configure vovserver to accept TLS 1.2 protocol by adding the following line to `SWD/policy.tcl`:

```
set config(http.minSSLVersion) "TLSv1.2"
```

3. **Optional:** Append a line to `SWD/setup.tcl` that sets `VOV_HOST_HTTP_NAME` to the fully qualified host name (FQHN) where vovserver is running. The FQHN is the output of `hostname -f`.


```
setenv VOV_HOST_HTTP_NAME FQHN
```

4. **Optional:** For a full HTTPS security, a CA-signed domain-wide SSL certificate is installed in `$VOVDIR/local/ssl` or a host-specific CA-signed SSL certificate is installed in `SWD/config/ssl`. This will allow your REST network traffic to be fully secured. See [Advanced REST Usage](#) for more about this.
5. Reread and activate the changed server configuration parameters via `vovproject reread` or similar commands.

Prerequisites for the REST client:

- The REST client must have network access to the server running the NC queue.
- Python version 3 or higher is required to use the `vov_rest_v3.py` REST access library module.

- Copy `vov_rest_v3.py` from `$VOVDIR/scripts/python/vov_rest_v3.py` to the directory on the REST client where your Python application resides.
- The following Python request packages must be installed: `json`, `nacl.utils`, and `urllib`

 **Note:** If key based authentication will be used, your VOV public key must be "added", or registered, with vovserver. See [Register Your Public Key on a vovserver Instance](#) for more about this.

Example Application: REST 101

Programs that use the Accelerator REST API can do so in two ways. The preferred approach is to utilize the Python library module "vov_rest_v3", which is provided with Accelerator software packages in the file `vov_rest_v3.py`. This Python module simplifies coding by hiding details of the HTTP operations that interface directly to the low level REST API. The `vov_rest_v3` module requires Python version 3 or higher.

For more advanced users, a later section of this guide shows how to interface directly to the REST API. Direct use can give you more control over the following:

- Management of JWT access tokens
- Policy of your application regarding insecure HTTPS configurations
- Control over the use of connection "keep-alive"


Direct REST API interface will be covered later. This approach works with Python version 1 and higher.

To get a quick start using REST, examine a simple "REST 101" example Python program that queries REST to return the name of an NC queue, shown below. The lines of the program are numbered and annotated with explanations.

```
1  #!/usr/bin/python3
2  #
3  # nc_rest_101.py
4  #
5  import os, sys, getpass, json
6  import vov_rest_v3
7
8  url = os.environ['NC_URL']
9
10 vrest = vov_rest_v3.VOVRestV3()
11 vrest.authorize(url, getpass.getpass('Password:'))
12 r = vrest.submitRequest("GET", url + "/api/v3/project/1/project")
13
14 print ( json.dumps(json.loads(r), indent=2) )
```

Here is a line-by-line guide to the above program.

Line 1 This Python program is compatible with Python 3.

- Line 6** Import the `vov_rest_v3` Python module that comes with the Accelerator product. Copy it locally from `$VOVDIR` before running the Python program.
- Line 8** Pull the URL for the NC queue out of an environment variable. The URL is what `nc cmd vovbrowser` shows.
- Line 10** Allocate `vrest`, an instantiation of the `VOVRestV3` Python class.
- Line 11** The `authorize()` method function authenticates using the password and allocates a JWT token stored in the object. This example obtains your user password by interactive prompt.
-  **Note:** This example uses username/password authentication. To use key-based authentication, see .
- Line 12** The HTTP get request is sent, returning a string containing the HTTP response data in JSON format.
- Line 14** The HTTP response string is parsed into JSON and pretty-printed.

Here is a terminal session that shows how to run the program.

```
% echo $NC_QUEUE
My_NC_QUEUE
% export NC_URL=`nc cmd vovbrowser`
% echo $NC_URL
http://myhost:6330
% cp $VOVDIR/scripts/python/vov_rest_v3.py .
% python3 nc_rest_101.py
Password:
{
  "startrow": 1,
  "endrow": 1,
  "query": "SELECT project FROM 1",
  "errmsg": "",
  "columns": [
    {
      "col": 0,
      "field": "project"
    }
  ],
  "rows": [
    [
      "My_NC_QUEUE"
    ]
  ]
}
```

nc_run.py

```
#!/usr/bin/python3
#
# nc_run.py
#
```

```
# Usage
#
#     export NC_URL=<URL FOR NC QUEUE>
#     ./nc_run.py <command> [args]
#

import os, sys, random, json, getpass
import vov_rest_v3

def getMyPassword():
    return getpass.getpass('Password:')

# Main body
nc_url = os.environ["NC_URL"]
scheme = nc_url.split(":")[0]
hostport = nc_url.split("/")[2]
url = "{0}://{1}".format(scheme, hostport)
command = " ".join(sys.argv[1:])

vrest = vov_rest_v3.VOVRestV3()
vrest.authorize(url, getMyPassword())

# Job attributes - required
VOV_JOB_DESC = {
    "command" : command,
    "logfile" : "JOBLOG." + str(random.randint(100000, 999999)),
    "rundir" : os.getcwd(),
    "env" : "BASE",
}

# Job attributes - optional / User specified
VOV_JOB_DESC.update( {
    "priority,sched" : 4,
} )

r = vrest.submitRequest("POST", url + "/api/v3/jobs", jsonData=VOV_JOB_DESC)
print ("New job is %s" % json.loads(r)["jobid"])
```

nc_list.py

```
#!/usr/bin/python3
#
# nc_list.py
#
# Usage
#
#     export NC_URL=<URL FOR NC QUEUE>
#     ./nc_list.py
#

import os, sys, json, getpass
import vov_rest_v3

def getMyPassword():
    return getpass.getpass('Password:')

def listJob(vr, url):
    query = ( url + '/api/v3/query'
              + '?select=id,statusnc,PRIORITYPP,host,command'
              + '&from=System:User:' + os.environ['USER'] )
    response = vr.submitRequest("GET", query)
    return response
```

```
def prettyPrint( text ):
    dd = json.loads(text)
    for ii in range(0, len(dd['columns']) ) :
        sys.stdout.write("%9.9s " % dd['columns'][ii]['field'])
    sys.stdout.write("\n")
    if ('rows' not in dd):
        return
    for rr in range (0, len(dd['rows']) ) :
        row = dd['rows'][rr]
        for ii in range(0, len(dd['columns']) ) :
            if (ii < len(dd['columns'])-1):
                sys.stdout.write("%9.9s " % str(row[ii]))
            else:
                sys.stdout.write("%10.30s" % str(row[ii]))
        sys.stdout.write("\n")

#
# Main body
#
nc_url = os.environ['NC_URL']
scheme = nc_url.split(":")[0]
hostport = nc_url.split("/")[2]
url = "{0}://{1}".format(scheme, hostport)

vrest = vov_rest_v3.VOVRestV3()
vrest.authorize(url, getMyPassword())
json_text = listJob(vrest, url)
prettyPrint(json_text)
```

Example Application: REST 101 Use Key Authentication

Your public VOV key must be registered with vovserver before running applications that use key based authentication.

A REST request can be programmed from Python by using method functions for the VOVRestV3 class in the vov_rest_v3.py module included in \$VOVDIR. Use the `authorizeWithKey()` function to authenticate, followed by the `submitRequest()` function to issue the REST request.

```
#
# nc_rest_101_vov_keys.py
#
import os, sys
import vov_rest_v3

url = os.environ['NC_URL']
secret_key = os.environ['MY_SECRET_KEY']
sp_key = os.environ['NC_SERVER_PUBLIC_KEY']
user = os.environ['USER']
vrest2 = vov_rest_v3.VOVRestV3()
vrest2.authorizeWithKey(url, secret_key, sp_key, username=user)
r = vrest2.submitRequest("GET", url + "/api/v3/project/1/project")
print (r)
```

Example Application: Job Submit and List

To demonstrate the utility of the REST API, here are two simple Python programs that imitate the function of the `nc run` and `nc list` commands that are familiar CLI tools from the Accelerator product. The source code for these tools, named `nc_run.py` and `nc_list.py`, is found on the following pages.

These REST programs only require that the `NC_URL` environment variable be set to the Accelerator queue URL, as returned by `nc cmd vovbrowser`. Here is a terminal image that demonstrates the simple REST tools.

```
% cp $VOVDIR/scripts/python/vov_rest_v3.py .
% export NC_URL="`nc cmd vovbrowser`"
% ./nc_run.py sleep 33
Password:
New job is 45080
% ./nc_run.py sleep 66
Password:
New job is 45083
% ./nc_list.py
Password:
```

ID	STATUSNC	PRIORITY	TYPE	HOST	COMMAND
000045080	Running	normal		myhost	vw sleep 33 > JOBLOG.185633
000045083	Running	normal		myhost	vw sleep 66 > JOBLOG.903723

Launch Jobs with Non-default Options

The `nc_run.py` example program shown earlier can submit an Accelerator job with no customizations. In reality, users often need to specify non-default properties for the jobs they submit. Examples are resources, slot counts, auto-kill times, and numerous other options that can be seen on the `nc run -h` CLI command help information and in the documentation with the *Altair Accelerator Administrator Guide*.

To view available job launch options, find the `VOV_JOB_DESC` table of options in of the *Altair Accelerator Administrator Guide*. Here is an excerpt of that table:

Field in Array	Description
autokill	Set the autokill flag (option -kill)
check,directory	Set it to 0 to disable checking of canonicalization of current directory (option -D)
env	Environment of the job (option -e). Set this to "" or to DEFAULT to force the use of an environment snapshot.
force	Force the job to be rescheduled (option -F)
group	Group the job belongs to (options -g and -G)

Field in Array	Description
inputs	List of input files (dependencies) (option -i)
priority, default	Default priority (NOT USED)

Although the above document has good descriptions of the fields/keys that can be provided to job launch requests, some of those fields are usable in Accelerator jobclass definitions but not in REST job requests. To see the precise list of fields/keys that are supported in REST requests, see the Swagger docs described in the [REST Request Detailed Documentation](#) section.

You can modify the `nc_run.py` script previously shown (or the `nc_run2.py` script in an upcoming section) with additional options specified as key-value pairs in the `VOV_JOB_DESC` Python dictionary. For example, here is the modification to `nc_run.py` that specifies an auto-kill time of 60 seconds. The Python dictionary "update" method function provides a convenient way to do that.

```
VOV_JOB_DESC.update( {
    "autokill"      : 60,
} )

vrest.submitRequest("POST", resturl, jsonData=VOV_JOB_DESC)
```

The vov_rest_v3.py Python Library Module

The Accelerator software package provides a Python library module called `vov_rest_v3.py` to make REST API usage from Python more convenient. The module implements a `VOVRestV3` Python class with member functions described in the following text box. These functions hide the details associated with authenticating, session handling, and error handling. More example programs follow that utilize this convenient Python library layer.

VOVRestV3 Python Class Description

The `VOVRestV3` Python class provides an interface to the Accelerator v3 REST API.

Location

```
$VOVDIR/./common/scripts/python
import vov_rest_v3
```

Member Functions

authorize (url, username="", password="")	The vovserver scheduling server authenticates a user for a <code>VOVRestV3</code> and obtains a validating access token behind the scenes, storing it in the <code>VOVRestV3</code> object.
Positional Arguments	<p><code>url</code> – (string) the URL as returned by <code>nc cmd vovbrowser</code></p> <p><code>password</code> -- (string) the current user password</p>

	Keyword Arguments	username – (string) user name, usually the same as the Linux user name known to vovserver. If this argument is not provided it defaults to the current user identified by \$USER
	Return Value	None. Raises exceptions upon failure.
	Submits a REST/ HTTP request to the server.	
submitRequest (method, url, queryParams={}, jsonData={})	Positional Arguments	method – (string) one of: "GET", "POST", "PUT", "DELETE", "PATCH". url – (string) the URL as returned by <code>nc cmd vovbrowser -url RESTPATH</code> . Examples of RESTPATH are: <code>/api/v3/jobs</code> or <code>/api/v3/projects/1</code> .
	Keyword Arguments	queryParams – (dictionary) keywords and values in string format for GET requests jsonData – (dictionary) keywords and values in string format for POST requests
	Return Value	(string) The HTTP request response text.
getJWT()	Retrieve the JSON Web Token (JWT) for the object. Return Value: (string) the JWT string	
setJWT(token)	Replace the JSON Web Token (JWT) for the object.	
	Positional Arguments	token – (string) the replacement JWT to be remembered in the object and used for subsequent REST requests
	Return Value	None.

REST Request Detailed Documentation

The Accelerator REST v3 API interface is described in detail under the Swagger documentation that comes with the Accelerator software. To browse the Swagger REST documentation, browse to the URL displayed as the output of this command:

```
nc cmd vovbrowser -url /html/vovrest.html
```

The information at this page will help you construct a valid REST request.

For example, browse on the Swagger documentation page to the “jobs” object. You will see that a POST request is used, and the REST URL segment is “/api/v3/jobs”.

jobs ▼		
POST	/jobs Create job	🔒
PUT	/jobs/{id} Job Control	🔒
PATCH	/jobs/{id} Modify job	🔒
DELETE	/jobs/{id} Removes job from server memory	🔒
GET	/jobs/{id} Query all fields on a single job.	🔒
GET	/jobs/{id}/{field} Query single field on a single job.	🔒

Figure 2: REST Object-Verb Reference

Click on the POST, PUT, PATCH, DELETE, or GET buttons for more details about the required parameters and supported dictionary keys and values for each request type. This gives you what you need to know to construct a complete and working job submit request via Python or curl.

Online REST Reference Pages

The Accelerator Web UI has a REST reference page complex. This page is an interactive VOV REST reference with a complete list of objects that may be targeted with REST accessors operations and a summary of job control functions that may target the “job” object.

To access the reference page, run the following:

```
% vovbrowser -url /html/vovrest.html
```

Issue REST Requests from the Swagger Web UI

The Swagger Rest API page provides a learning interface that helps you formulate and execute well-formed REST requests to the associated NC queue. The web UI page shows a construction of the REST request and response text in curl command and JSON language formats.

Here are the steps to formulate and execute a REST request:

1. Access the VOV REST API page in a browser. The URL is shown by the command:

```
nc cmd vovbrowser -url /html/vovrest.html
```

2. If SSL/TLS is enabled for the NC queue, then select the https URL in the "Server" drop-down menu on the left side of the page near the top.
3. Scroll down to the Object-Verb section for the desired REST request.
4. Click on **Try it Out** on the right.
This activates a web form on the page to accept your specified parameters for a REST request before executing it.
5. Edit the parameters for your REST request in the web form.

Example A

If you are selecting a GET operation on a job object, the job id and sometimes a job object field name are required. Enter these items in the web form.

Example B

If you are selecting a POST operation on a job object to submit a job to Accelerator, the parameters are more extensive. Edit the job creation parameters box. A template window in the web form illustrates the format, which is consistent with a Python dictionary syntax with keyword/value pairs. Replace the template with your desired job creation specifications.

Most of the keyword/value pairs in the template for job creation are optional, but a few are required. A simple and minimal working example job creation parameter list follows. To specify other properties of a job, choose some additional keywords from the template text that appears pre-populated in this sub-window.

```
{  
    "command" : "sleep 60",  
    "logfile" : "JOBLOG.01",  
    "rundir"  : "/tmp",  
    "env"     : "BASE"  
}
```

6. Click the blue **Execute** button.
This sends the REST request.
7. Scroll down to see the Server Response. A code of 200 - 299 indicates success.
8. Troubleshoot if the request failed. Common errors and remedies include:
 1. If error text is "Bad Request", then check that the right URL is selected in the Servers drop-down menu at the upper left.
 2. If an authentication error is seen, logout of the web UI and then log in again so the Swagger page gets a fresh JWT access token.
 3. If an error with "Error: Bad Request" is seen, check the specified text in the Job Control Parameters sub-window. A common mistake is to add an invalid comma after the last keyword/value pair.
 4. A server error is returned if you specify a job logfile that is the same as another job in the system. Change the logfile name and retry the request.

Advanced REST Usage

This section will cover some more advanced REST API topics: management of JWT access tokens, HTTPS secure and insecure connections, and connection keep-alive. The example application, `nc_info.py`, imitates the familiar Accelerator CLI command `nc info`, which returns information about a job. This application will interface to the REST API directly, instead of using the `vov_rest_v3` module interface layer.

This REST application requires the `NC_URL` environment variable to be set to the Accelerator queue URL, as returned by `nc cmd vovbrowser`. Also, this Python program needs JWT-handling module `getToken.py` from the next section in this guide. Create that python file before running `nc_info.py`.

Here is a shell session that shows how to run `nc_info.py`. In this example, two jobs are started, and an invocation of `nc_info.py` will query and display information about the two running NC jobs.

```
% nc run -v 2 sleep 123
Job <000001138> is submitted
% nc run -v 2 sleep 456
Job <000001143> is submitted
% export NC_URL=`nc cmd vovbrowser`
% echo I will need `ls getToken.py`
I will need getToken.py
% ./nc_info.py 000001138 000001143
Password:
Job          000001138
User,Group   user99,/time/users.user99
Command      vw sleep 123 > vnc_logs/20211012/132628.122875
Status       Running
  Host       myhost
  Duration   31s

Job          000001143
User,Group   user99,/time/users.user99
Command      vw sleep 456 > vnc_logs/20211012/132633.122899
Status       Running
  Host       myhost
  Duration   26s
```

nc_info.py

```
#!/usr/bin/python
#
# nc_info.py
#
# Usage
#
#     export NC_URL=<URL FOR NC QUEUE>
#     ./nc_info.py JOB_ID [JOB_ID ...]
#

import os, sys, json, requests
from getToken import getJWT

def getMyPassword():
    import getpass
    return getpass.getpass('Password:')

def infoPrint( text ):
    dd = json.loads(text)
    id   = find(dd, 'ID')
    user = find(dd, 'USER')
    group = find(dd, 'GROUP')

    print ("%16s" % ("Job", id) )
    print ("%16s,%s" % ("User,Group", user, group))
    print ("%16s" % ("Command", find(dd, 'COMMAND')))
    print ("%16s" % ("Status", find(dd, 'STATUSNC')))
    print ("%16s" % ("  Host", find(dd, 'HOST')))
    print ("%16s" % ("  Duration", find(dd, 'DURATIONPP')))

def find(jobdump, key):
    for c in range (0, len(jobdump['columns'])):
        if (jobdump['columns'][c]['field'] == key):
            break
    return jobdump['rows'][0][c]

#
# Main body
#
nc_url = os.environ['NC_URL']
scheme = nc_url.split(":")[0]
hostport = nc_url.split("/")[2]
url = "{0}://{1}".format(scheme, hostport)

ss = requests.Session() # use keep-alive
jwt = getJWT(ss, url, os.environ["USER"], getMyPassword())
for arg in range (1,len(sys.argv)):
    jobid = sys.argv[arg]
    query = url + '/api/v3/jobs/' + jobid
    r = ss.get(query, headers={"Authorization": jwt},
verify=True)
    infoPrint(r.text)
    print ("")
```

JWT Access Tokens

To authorize REST access via the API, REST requests must pass an access token in the request header. In the `nc_info.py` example, the `jwt` variable holds the access token. Access tokens expire about 4 hours after issue. An application that will run for several hours should adopt some strategy to allow for access token expiry. One strategy would be to re-authenticate, using login name and password, prior to any burst of REST activity that will be known to be complete in a few hours or less. Another strategy is to check the request return status and re-authenticate at that time, using the new access token thereafter.

The recommended way to authenticate user name and password to issue an access token is the VOVRestV3 Python class `authorize()` method function. If you would like direct control over the access token allocation, as in the `nc_info.py` example, see the `getJWT()` function in the `getToken.py` example in the next section.

HTTPS Security Considerations

REST requests and responses are sent over the HTTP communication protocol, and suitable HTTP connections can be configured with three possible security levels, ranging from insecure to very secure:

1. The basic (insecure) way to start the Accelerator queue server and web server is to use the default HTTP connection on the server's webport. The REST API interface always is allowed on HTTP webport connections.
2. An intermediate level of security is possible if the HTTPS protocol is requested by configuring server parameter `ssl.enable` to 1 in `policy.tcl`. In this case, a self-signed SSL certificate will be generated when the Accelerator server starts. If the REST application uses the VOVRestV3 python module method functions, this type of connection will be supported without warnings being issued. If direct access to the REST API is implemented using the Python request method functions `get()` and `post()`, then communication will be allowed on these connections only if the optional `verify=False` keyword argument is passed to those functions.
3. A very secure HTTPS connection will be established if an SSL certificate that was signed by a trusted Certificate Authority (CA) is added to the Accelerator server configuration. This is the recommended way to configure Accelerator products. If the REST application author would like to require level 3 security, then the direct access to REST via request method functions `get()` and `post()` must be used, and the keyword argument `verify=True` must be specified.

Connection Keep-Alive

The use of HTTP keep-alive, or persistent connection, is an important technique that optimizes applications during times of very frequent REST requests. HTTP keep-alive and the resultant reuse of HTTP connections will occur when using the VOVRestV3 `submitRequest()` method function or when using a "session" allocated by the `request.Session()` method function.

The `nc_info.py` Python application illustrates the use of keep-alive by the latter method in its main loop across job IDs. Each call to `ss.get()` will reuse the same HTTP connection. If the `ss.get()` call were to be replaced by `request.get()`, the keep-alive feature would not persist an HTTP connection

between subsequent requests. In that case, each request would build a new HTTP connection before issuing the request.

JWT Token Allocation

Accelerator REST API v3 uses JSON Web Tokens (JWTs) to implement the access tokens used in REST requests. A JWT access token is allocated by an HTTP POST request by the client that passes a matching Linux username and password along with an implementation-defined REST URL. The server responds by verifying the username and password match and returning the JWT access token to the client.

The `authorize()` method function in the `VOVRest` Python module should be used for most convenient authorization and automatic JWT handling. This method is used in the later examples in this tutorial. The `getToken.py` module that follows shows the low level interface to JWT allocation. This module must be provided with the first few working Python code examples shown in this tutorial.

getToken.py

```
# JWT Token utilities

import os, requests

#
# Function getJWT()
#
# Arguments
#
#     url          - A URL for a VOV project
#     user         - user name for authentication
#     password     - password for authentication
#
def getJWT(url, username, password):
    scheme = url.split(":")[0]
    hostport = url.split("/")[2]
    baseUrl = "{0}://{1}".format(scheme, hostport)
    tokenUrl = baseUrl + "/api/v3/token"
    myauth={ 'username' : username , 'password' : password }
    r = requests.post(tokenUrl, data=myauth)
    if ( r.status_code > 300 ):
        print ("JWT Error code %d" % r.status_code)
        print ("          returned status: ", r.json() )
        print ("          error message : %s" % r.json()['error'])
        exit(1)
    token = r.json()
    jwtToken = token['token_type'] + ":" + token['access_token']
    return jwtToken

#
# Function getMyPassword()
#
# This example function simply prompts the user to type the account password.
#
def getMyPassword():
    import getpass
```



```
return getpass.getpass('Password:')
```

The Python code in the **getToken.py** module also contains a placeholder password prompt function **getMyPassword()**. The handling of user passwords and JWT tokens in practice will be up to the REST application developers in accordance with their own best practices for handling and storing security-sensitive information. If the REST application runs for many hours, new JWT tokens will need to be allocated and authenticated after the previous ones expire. The application needs to provide a way to provide the password each time a JWT token is authenticated. Additional methods for renewing or allocating JWT tokens are being considered for future Accelerator software releases.

All About Tcl

Tcl Syntax

1. Every script is a sequence of commands separated by newlines or semicolons.
2. All commands are formed by one or more words separated by spaces or tabs.

```
CMD arg arg arg ...
```

3. The first word is always the **name** of the command.
4. The remaining words are the **arguments**.
5. Even ``if'', ``while'', ``for'', ``foreach'' and ``switch'', and the definition of new procedures obey this rule.
6. If the first character of a command is a ``#' the text that follows (up to the next newline) is treated as a comment.

Example:

```
% set a 10
% puts "This is the value of a: $a "
% while {$a > 0} { puts $a }
% if { $a == 0 } { puts "a=0" } else { puts "a!=0" }
```

Tcl Special Symbols

Table 3:

Symbol		
\$	Dollar sign	Used for variable substitution. This substitution is executed inside the double quotes and the square brackets, but NOT inside the curly braces. Example: <div> <pre>% set my_message "Hello!" % puts "I'd like to say : \$my_message"</pre> </div>
[]	Square brackets	Same meaning as of the reverse-quote (``) for the UNIX shells. The string enclosed in brackets is replaced with the result of the execution of the string itself. Example: <div> <pre>% set a [expr 10 + 10]</pre> </div> <p>In the example, the command ``expr'' evaluates the three arguments as a mathematical expression. The</p>

Symbol		
		resulting string becomes the second eargument of the ``set" command.
\	Backslash	The backslash can be used to access special characters (\\$, \[, \]).
""	Double quote	<p>Group several characters into a single argument. Necessary every time an argument contains spaces, tabs or newlines.</p> <p>Example: print a string with an embedded newline:</p> <pre>% puts "Hello world! This is a wonderful day" Hello world! This is a wonderful day</pre>
{ }	Curly braces	<p>Group words into a single argument. Elements within the braces are not interpreted, so no variable or command substitution takes place.</p> <p>Example:</p> <pre>% set HOME "/home/username" % puts {This won't be interpreted: \$HOME } This won't be interpreted: \$HOME % puts "While this will be substituted: \$HOME" While this will be substituted: /home/ username</pre>

Simple Variable

Tcl provides a typeless kind of variable that can be used to store strings, numbers, lists and scripts:

- Always stored as strings
- No need for declarations
- Created automatically when a value is assigned...
- ... and removed with the ``unset" command
- Variable names are case sensitive
- The value is assigned with the ``set" command ...
- ... and accessed using the dollar notation (\$)

Examples:

```
% set a {this is a string}
% set n 9.1 ;# this is a string too !
```

```
% unset a n ;# unset the two variables
```

When a variable name has to be concatenated with a letter, a number or an underscore (without spaces in between), the variable name has to be quoted with a pair of curly braces.

Example:

```
% set temperature 12.3
% puts ${temperature}F
12.3F
```

Associate Arrays

Tcl provides arrays, in the form of collection of elements. The complete name of an array element has two parts:

1. The name of the array
2. The name of the element within the array

Examples:

```
% set monthDays(January) 31
% set monthDays(February) 28
% set dummy(this-is-the entry-name) {any value, even $HOME}
# note the dashes and the space between 'the' and 'entry'
```

To access an element use the dollar notation:

```
% set Year(Jan) 31
% puts $Year(Jan)
```

But when the name of the element is a variable itself a double substitution is performed:

```
% set anArray(theElement) "the value"
% set foo "theElement"
% puts $anArray($foo)
```

Spaces and commas in the element name are treated as part of the name.

Simulate multidimensional arrays, by concatenating multiple indices into a single element name.

Examples:

```
% set calendar(monday,0800) {Breakfast with John}
% set calendar(sunday,1100) {Game: go bears!}
% set day sunday
% set time 110
% puts $calendar($day,$time)
Game: go bears! ;# output
```

The comma (,) is only a common way to separate the components of the element name. Every symbol between parentheses is interpreted as part of the element name. Do not add spaces!

Lists

A list is an ordered collection of elements. It is represented as a string with a particular structure. The simplest form of list is a sequence of strings separated by spaces. Lists can be nested, using curly braces as delimiters. The empty string is {}

Example:

```
% set L1 "a b c d"
% set L2 { 1 { a b } 2 { hello "a b" } }
% lindex $L2 1
a b
% lindex $L2 3
hello "a b"
% lindex [ lindex $L2 3 ] 1
a b
% lindex [ lindex [ lindex $L2 3 ] 0 ]
a
% lindex $L2 end
hello "a b"
```

The example shows the use of the `lindex` command to access the elements of a list.

- `lindex` wants two (2) arguments: a list and a position
- The index 0 corresponds to the first element
- The "end" symbol corresponds to the last element
- The elements of a nested list (like L2) can be accessed using nested invocation of the `lindex` command (see: command substitution via square brackets)

Procedures

- The definition of a new procedure is done with a command, called `proc`
- It takes three (3) arguments: name, list of arguments, and body: `proc name args body`

Example:

```
% proc Fact { n } {
    if { $n <= 1 } {
        return 1
    }
    return [expr $n * [Fact [expr $n-1]]]
}

% Fact 4
24
```

- More like functions in C, rather than Perl
- Variables created inside a procedure are local to that procedure
- To access global variables, use global keyword
- Take arguments as any other Tcl command (i.e. space separated)
- Return a string with the return command
- The argument list is actually a list of lists...
- ... each sublist has either one or two elements
- In the latter case, the second argument is the default value

Example:

```
proc my_inc { value {increment 1} } {  
    return [expr $value + $increment]  
}
```

Tcl Primitives

Lists

concat list [list ...]

Concatenates multiple lists into a single lists. Each element of each list become an element of the new one.

Returns the list.

Example:

```
% set L1 {a b {c d} e}  
% set L2 {f g {h i}}  
% concat $L1 $L2  
a b {c d} e f g {h i}
```

join list [joinString]

Joins the elements of a list into one single string using joinString as separator (default: one space).

Returns the string.

Example:

```
% set L1 { {} home username public_html rtda {} }  
% join L1 /  
/home/username/public_html/rtda/  
# note what happened to the two {}
```

lappend list value [value ...]

Appends each value to the current list as a new list element.

Returns the list.

Creates a new list if it doesn't exist.

Example:

```
% set fullpath { {} home }  
% set user { username }  
% set folder { public_html rtda {} }  
% lappend fullpath $user $folder  
{ } home { username } { public_html rtda {} }
```

lindex list index

Returns the index'th element of the list. The first element has index 0. The last can be accessed using the symbol ``end".

list [value ...]

Concatenates the arguments to make a new list. Each argument becomes an element of the new list.

Returns the list.

Example:

```
% set L1 {a b {c d} e}
% set L2 {f g {h i}}
% list $L1 $L2
{a b {c d} e} {f g {h i}}
```

llength list

Returns the number of elements in the list.

split string [splitChars]

Returns a list formed by splitting string at instances of `splitChars` and turning the characters between these instances into list elements.

Example:

```
% set fullPath "/home/username/public_html/rtda/read.me"
% split $fullPath /
{} home username public_html rtda read.me      ;# note the {}
```

Expressions

- Almost identical to C/C++
- Evaluated through the `expr` command
- In the `if` and `while` clause the `expr` is implicit

Example:

```
% set x .123
% expr 2*sin($x)
```

- Some operators (`<`, `>`, `<=`, `>=`, `==`, `!=`) can be used with `string` in the `if` statement

Example:

```
% set s "hello"
% if { "hello" == $s } { do_something }
```

- If both the operands of a `$==$` are numerical Tcl uses numeric comparison (see string operation for the string compare command).

Pattern Matching

Used in the following commands: `glob`, `info`, `lsearch`, `regexp`, `regsub`, `string match`, `switch`.

Tcl provides two (2) different techniques:

1. Glob (aka wild cards)

2. Regular expressions

Glob

- Easier to use than the regular expression
- Works well on simple cases
- Used by `glob`, `info` and `string match`
- Used through the `-glob` option by `lsearch`, `switch`

<code>*</code>	Matches any sequence of zero or more characters
<code>?</code>	Matches any single character
<code>[char]</code>	Matches any single character in chars. A sequence of chars can be defined using the dash symbol (i.e. a-z, 0-9)
<code>\x</code>	Matches the single special character x. Used to avoid ambiguity with the other symbols.

Regular Expression

- More powerful technique for pattern matching
- Same syntax of `grep -e`, `egrep`
- Subset of the one implemented by Perl
- Used by `regexp` and `regsub`
- Used through the `-regexp` option by `lsearch` and `switch`
- The basic building block of a pattern is called **atom**

Atom	Meaning
<code>.</code>	Matches any single character
<code>^</code>	Matches the null string at the beginning of the input string
<code>\$</code>	Matches the null string at the end of the input string
<code>\x</code>	Matches the character x
<code>[chars]</code>	Matches any single character in chars. If the first character is <code>^</code> , then it matches any character that is NOT in the set. If the first character (possibly after <code>^</code>) is <code>]</code> then it is treated literally. If <code>-</code> is the first, it is treated literally.
<code>(regexp)</code>	Matches anything that matches the <code>regexp</code> . Used for grouping and for identifying pieces of the matching substring.

Operator	Meaning
*	Matches any sequence of zero or more matches of the preceding atom
+	Matches any sequence of one or more matches of the preceding atom
?	Matches either the null string or a match of the preceding atom
regexp1 regexp2	Matches anything that matches either <code>regexp1</code> or <code>regexp2</code>

String Manipulation

format formatString [value ...]

Similar to the C function `sprintf`. The return value is the `formatString` with the `%` sequences replaced with the values.

Example:

```
% set tot 123
% puts [format "Total amount: %4d" $tot]
```

regexp [-indices] [-nocase] [--] expr string [matchVar] [subVar ...]

Returns 1 if the regular expression `expr` matches the string, 0 if it doesn't. `matchVar` is filled with the substring that matches the entire expression; the `subVar` variables, instead, receive the strings that match the subexpression (defined with the parentheses).

Example:

```
% puts "match: $m, sub: $s" ;# s if empty
match: .c, sub:
% regexp {.*/(.*)\.(.*)$} "/home/username/emacs.rc" m n e
1
% puts "match: $m, name: $n, ext: $e"
match: /home/username/emacs.rc, name: emacs, ext: rc
```

- Use `-nocase` for a case insensitive matching
- Use `--` to say that the following argument is the expression even if it starts with a dash `--` (i.e. it's not an option)

regsub [-all] [-nocase] [--] expr string subSpec resultStr

As for `regexp`, a pattern matching on the string is performed using `expr`. The matching substrings are then replaced with the `subSpec` string and stored into `resultStr`.

Example:

```
% regsub , "a,b,c,d,e" \t res
```

```
1
% puts $res
a   b,c,d,e
% regsub -all , "a,b,c,d,e" \t res
4
% puts $res
a   b   c   d   e
```

- If the `-all` option is not selected only the first occurrence is replaced.
- The return value is the number of matching substrings.
- The `-nocase` and `--` options work as for `regexp`

scan string format varName [varName ...]

Similar to the C function `sscanf`. Parses the fields of the string according to the format string. Each `%` sequence is stored in the corresponding `varName`. Returns the number of fields successfully parsed.

Example:

```
% set line "123 days, 12 hours"
% set form "%d days, %d hours"
% scan $line $form d h           ;# d gets 123, h gets 12
```

string compare string1 string2

As for `strcmp`, returns

- -1 if `$string1 < $string2`
- 0 if `$string1 == $string2`
- +1 if `$string1 > $string2`

Control Flow

- The structures are similar to C and UNIX `csh` shell
- They are implemented as commands, as any other thing in Tcl
- Remember: Tcl needs a space to tell one argument from another
- Braces are NOT separators (as in C and Perl)

for init test reinit body

- Similar to the C and Perl implementation
- All the arguments are normal Tcl scripts
- The `for` command returns an empty string as the result

Example:

```
% for {set i 1} {$i<=10} {incr i} { puts -nonewline "$i," }
1,2,3,4,5,6,7,8,9,10,
% set n 5
% for {set res 1; set i $n} {$i>1} {incr i -1} {
    set res [expr $res*$i]
}
% puts $res
```

```
120                                     ;# 5! = 120
```

foreach varName list body

- Basic form
- Iterates over all the elements of the list
- While executing the body, the current element is stored in the variable `varName`

Example:

```
% set cirFiles {bjtnoise mos6inv rca3040 schmitt}
% foreach c $cirFiles {
    runspice $c.cir
}
```

foreach varList1 valList1 [varList2 valList2 ...] body

See TCL book for a full explanation.

while test body

- Evaluate the test script as an expression
- Executes the body script if it's nonzero and then ...
- ... re-evaluate the test and loop
- Returns an empty string

Example:

```
set done 0
while {$done} {                                ;# "while \{!$done\}\{" is wrong!
    set done [do_task]                          ;# Put a space between \} and \{
}

set fp [open $filename]
while { [gets $fp line] != -1 } {
    puts "I've read : $line"
}
```

if test1 body1 [elseif test2 body2 ... else bodyn]

- Evaluate the test1 script as an expression
- If it's nonzero executes the body1 script and returns its value, otherwise ...
- ... evaluates the test2 script
- ... and so on...

Example:

```
if { $name != $entryname } {                    ;# note the != between strings
    vhs_parse_to_endl skip $fp
} else {                                         ;# remember the spaces!
    vhs_parse_to_endl doit $fp
}
```

switch [options] string { pattern body [pattern body...] }

- Matches string against each pattern in order, until a match is found
- Executes the corresponding body and returns its value

- Doesn't keep on matching, as in C/C++
- If the last pattern is ``default'', it matches any value
- If a body is – then Tcl uses the body of the next pattern
- Options are -exact, -glob, -regexp and --
- Use -- to tell that the next argument is the string, even if it begins with a dash (-)

Example:

```
switch -- $arg {
    "-d" { VovOutput "$FILEPREFIX.tab.h" }
    "-l" -
    "-r" {}
    "-b" { set FILEPREFIX [shift] }
    "-v" {
        if { [file tail $argv0] == "byacc" } {
            VovOutput "$FILEPREFIX.code.c"
        } else {
            VovOutput "$FILEPREFIX.output"
        }
    }
    default {
        VovFatalError "Unknown option $arg"
    }
}
```

I/O

- Syntax and behavior are similar to the C I/O library
- UNIX-like commands: `cd`, `pwd`
- File handling: `open`, `close`
- I/O : `gets`, `puts`
- Information and misc: `eof`, `file`, `glob`
- Pre-defined standard files: `stdin`, `stdout`, `stderr`

cd [*dirName*]

Usual UNIX syntax

pwd

Usual UNIX syntax

open *fileName* [*accessType*]

Opens the file named *fileName*

Returns a file identifier (the `fileId` of `gets`, `puts`,...)

accessType could be

- | | |
|-----------|--------------------------------------|
| r | Read Only. File must exist. Default. |
| r+ | Read and Write. File must exist. |

w	Write Only. Truncates the file if already exists. Otherwise creates a new one.
w+	Write and Read. Truncates if exists. Otherwise creates a new one.
a	Write only. Append if exists. Otherwise creates a new one.
a+	Write and Read. Append if exists. Otherwise creates a new one.

Example:

```
% set fp [open $fname r]
```

close fileId

Close the file given by fileId and returns an empty string.

Example:

```
close $fp
```

gets fileId [varName]

Reads the next line from the file

Stores the line into `varName`

Returns the number of characters, or -1 for end of file (eof)

If `varName` is not provided, returns the line or an empty string for end of file

Removes the terminating newline

Example:

```
set fp [open $fname r]
while { [gets $fp line] >= 0 } {
    parse_line $line
}
gets stdin a                                ;# reads from stdin
```

puts [-nonewline] [fileId] string

Writes string into `fileId`

If `fileId` is not provided, `stdout` is assumed

Appends a newline unless the `-nonewline` option is used

Returns an empty string

eof fileId

Returns 1 if an end-of-file condition has occurred on `fileId`. 0 otherwise.

file { exists|rootname|dirname|tail|extension ...} args ...

A family of commands to get information about files and directories.

Example:

```
set setupName setup.ini
if { [file exists $setupName] } {
    read_setup $setupName
}
set extension [file extension $setupName]    ;# gets ".ini"
```

glob [-nocomplain] [--] pattern [pattern ...]

Return a list of files that match any of the patterns

Uses `glob` pattern matching technique

If `-nocomplain` isn't specified, an error occurs if nothing matches

Use `-` to say that the next argument is the pattern (if it begins with a dash)

Example:

```
set DESIGN dummy
foreach sfx { sp st* pa* lis tr* mt* ac* ma* sw* ms* } {
    foreach file [glob -nocomplain "$DESIGN.$sfx"] {
        VovInput $file
    }
}
```

Command Line Arguments

- The command line arguments are stored in a list called `argv`.
- The name of the command IS NOT the first element of the list...
- ... instead it is stored in the variable `argv0`.
- The number of elements is stored in `argc`.

Tcl VOV Extensions

The standard Tcl interpreter is extended with some procedures, defined in the file `vovutils.tcl`.

Procedure	Description
<code>arglength</code>	Returns the length of the global variable <code>argv</code>
<code>lappend_no_dup list [var ...]</code>	<ul style="list-style-type: none"> • A variation on <code>lappend</code> that prevents duplicates • The arguments can be lists • Returns the new list <p>Example:</p> <pre>lappend_no_dup make(dirlist,include) ../include</pre>
<code>lprepend_no_dup list [var ...]</code>	Similar to <code>lappend_no_dup</code> , except the new elements are added at the beginning of the list
<code>shift [list]</code>	<ul style="list-style-type: none"> • Similar to <code>shift</code> in <code>csh</code> and in <code>Perl</code>

Procedure	Description
	<ul style="list-style-type: none"> The input list is passed by reference (i.e. is modified by the procedure) By default the top level <code>argv</code> list is used Returns the first element in the input list, or the null string if the input list is empty <p>Example:</p> <pre>while [arglength] { set arg [shift] VovOutput \$DESIGN.sav/\$arg.cfg }</pre>

Tcl Procedures (Alphabetical)

A

A

File	<code>vovfdl.tcl</code>
Usage	<code>A note check (default=1)</code>
Description	Add a note to the most recently mentioned node, that is the node with id <code>make(lastNodeId)</code> . The first argument is the text of the note. If the second argument, <code>check</code> , is set to 1, the procedure checks all notes to avoid duplicates.
Other procedures Used	<code>vtk_annotation_add</code>
Return	Nothing

AD

File	<code>vovfdl.tcl</code>
Usage	<code>AD job1 jobOrSet2 barrier (default=0)</code>
Description	This procedure is for those who want to have a dependency between jobs even if there is no real file-based dependency. Create an artificial dependency between <code>job1</code> and <code>job2</code> using a phantom place (see docs about phantoms) In large flows, it is

possible that this AD is called on job ids that are already executed and forgotten. Gently handle those errors but continue.

Return

The ID of the phantom place.

addComponent

File

vovcgi.tcl

Usage

addComponent name args

Description

Adds a component to a page in the web UI

addJsFile

File

vovcgi.tcl

Usage

addJsFile jsFile

addMenu

File

vovcgi.tcl

Usage

addMenu name url (default=) args

Description

Adds a menu to the current tab

addPage

File

vovcgi.tcl

Usage

addPage name url args

Description

Adds a page to the current menu

addTab

File

vovcgi.tcl

Usage

addTab name url args

Description

Adds a tab to the web UI

ADfast

File

vovfdl.tcl

Usage

ADfast job1 job2 args

AJAX_ELEM

File	vovhtml.tcl
Usage	AJAX_ELEM id message (default=) img (default=)

appendPageMessage

File	vovhtml.tcl
Usage	appendPageMessage msglevel msg

arglength

File	vovutils.tcl
Usage	arglength
Description	Return the length of the global argv variable.

AutoAjaxReloadNode

File	vovhtml.tcl
Usage	AutoAjaxReloadNode id refresh (default=4000)
Description	Automatically reload the page if the status of node changes.

AutoAjaxReloadSet

File	vovhtml.tcl
Usage	AutoAjaxReloadSet id refresh (default=4000)
Description	Automatically reload the page if the status of the set changes.

B

barrierClose

File	vovutils.tcl
Usage	barrierClose fp

BODY

File	vovhtml.tcl
Usage	BODY args

BR

File vovhtml.tcl

Usage BR

buildArgSpec

File vovhtml.tcl

Usage buildArgSpec what value

Description Args: Parameter name and a list of values

Returns Filter string to be used in an eval call to a VovLM report proc.

buildUrlParameters

File vovhtml.tcl

Usage buildUrlParameters what value

Description Args: Parameter name and a list of values

Returns URL fragment consisting of a parameter/value pair for each value

BUTTON

File vovhtml.tcl

Usage BUTTON url label options (default=)

C

calculateEndTime

File resplots_lib.tcl

Usage calculateEndTime start interval count

CAPSULE

File vovfdl.tcl

Usage CAPSULE args

Description Create a capsule on-the-fly for the most recent job declared with a T or J.

Usage CAPSULE [OPTIONS]
ENCAPSULATION_SCRIPT

Options	-vars "list_of_vars_to_be_passed"
Returns	OK
Side Effects	The script is evaluated, and a property CAPSULE is created.

Example

```
CAPSULE { I aa; O bb }
CAPSULE -vars [list cell corner]
{
  I $cell.$corner.in; O $cell.
  $corner.out
}
```

checkAccess

File	vovcgi.tcl
Usage	checkAccess allowedSecNum allowedUserList allowedGroupList
Description	Checks to see if the current user has access rights to a supported VovCGI object per its access rule

checkComponentSecurity

File	vovcgi.tcl
Usage	checkComponentSecurity component
Description	Processes the security check for web page components.

checkNavSecurity

File	vovcgi.tcl
Usage	checkNavSecurity tab menu (default=) page (default=)
Description	Processes the security checks for tab, menu, and page. Security settings cascade, so the parent's settings are always honored. Tab is always required, menu and page are optional.

COLOR

File	vovhtml.tcl
Usage	COLOR color text

ConfigureFeature

File vovhealthlib.tcl

Usage ConfigureFeature feature args

Example

```
ConfigureFeature FEATURE OPTION VALUE
```

ConfigureTag

File vovhealthlib.tcl

Usage ConfigureTag tag args

Example

```
ConfigureTag TAG OPTION VALUE
```

createSpanId

File vovhtml.tcl

Usage createSpanId spanId

Description Return a JavaScript-legal identifier string

D

D

File vovfdl.tcl

Usage D db

Description Specify database for places (default is FILE).

Returns Nothing.

Side Effects make (db) is set.

DefineEquivalence

File vovutils.tcl

Usage DefineEquivalence logical physical

DefineGroup

File vovutils.tcl

Usage DefineGroup name weight users

DefineTasker

File vovutils.tcl

Usage DefineTasker host type rshcmd (default=rsh)

DefineTaskerWithArgs

File vovutils.tcl

Usage DefineTaskerWithArgs host type args

deleteAppropriateWindow

File vov_common.tcl

Usage deleteAppropriateWindow win

DEPRECATED_OLD_vovMessage

File vov_common.tcl

Usage DEPRECATED_OLD_vovMessage title msg rows (default=10)
cols (default=50) font (default=)

Description Popup a text window to display the given title and message.

Return The message window.

dictEquals

File vovutils.tcl

Usage dictEquals d1 d2

doTestHealthFailoverServerCandidates

File vovhealthlib.tcl

Usage doTestHealthFailoverServerCandidates args

doTestHealthJobsWaitingDefaultJobclass

File vovhealthlib.tcl

Usage doTestHealthJobsWaitingDefaultJobclass args

doTestHealthServerDiskSpace

File vovhealthlib.tcl

Usage	<code>doTestHealthServerDiskSpace args</code>
Description	Check for low free space on SWD disk

E-F

E

File	<code>vovfdl.tcl</code>
Usage	<code>E newEnv script (default=)</code>
Description	Switch environment for all subsequent transitions.
Side Effects	<code>make (env)</code> and <code>env (VOV_ENV)</code> are set.

FDL_INIT

File	<code>vovfdl.tcl</code>
Usage	<code>FDL_INIT</code>
Description	Procedure that can be called to make sure that FDL is ready. In this case, there is nothing to do other than sourcing this file.

findDocPage

File	<code>vovcgi.tcl</code>
Usage	<code>findDocPage htmlPage</code>

FLAGS

File	<code>vovfdl.tcl</code>
Usage	<code>FLAGS args</code>
Description	Apply flags to the most recent transition.

FORM

File	<code>vovhtml.tcl</code>
Usage	<code>FORM args</code>

fossil

File	<code>vovutils.tcl</code>
Usage	<code>fossil args</code>

FSGROUP

File	<code>vovfairsharelib.tcl</code>
Usage	<code>FSGROUP group args</code>
Description	<p>Usage: <code>FSGROUP <group> [OPTIONS]</code></p> <p>Where <code>OPTIONS</code> can be:</p> <ul style="list-style-type: none">• <code>-user</code>: This is a USER level group• <code>-w Weight</code>: Set weight (default is inherited from parent)• <code>-t TimeWindow</code>: Set time window (default is inherited from parent)• <code>-f <0 1></code>: Set the flatten flag (default is 0)• <code>-u UserList</code>: Users who have access to the group.• <code>-l Limit</code>: Set limit for the resource associated with this group

FTButton

File	<code>vovhtml.tcl</code>
Usage	<code>FTButton url label extra (default=) jsCondition (default=)</code>

G

genControlFormComboMSelect

File	<code>vovcgi.tcl</code>
Usage	<code>genControlFormComboMSelect cgiScript name show size selections upstream options</code>

genFormHiddenInputsFromOptions

File	<code>vovcgi.tcl</code>
Usage	<code>genFormHiddenInputsFromOptions excludeList (default=) doInputs (default=1)</code>
Description	<p>Frequently, we need to generate a form that upon submit, will get the URL again just with 1 or 2 options changed, while keeping the other options unchanged. This procedure will get all the options from array <code>opt</code>, except ones specified in <code>excludeList</code>, and generate HTML code of a hidden input for each option.</p> <p>ARGS:</p>

- `excludeList` -- the list of option names to be excluded for generation of hidden input
- `doInputs` -- if set to 0, just return the query string without generating hidden inputs

Returns

Returns the part of query string of the options that hidden inputs are generated. For example:

```
aa=vaa&bb=vbb&cc=vcc
```

Example

A form that just changes the username option but keeps all other options as they are.

```
FORM { INPUT text username $opt(username)  
  VovCGI::genFormHiddenInputsFromOptions username }
```

genReportHeader

File

`vovcgi.tcl`

Usage

```
genReportHeader showTimeRange (default=1) showTag  
(default=1) showFeature (default=1)
```

Description

Currently used by Monitor only.

Inserts a fieldset that contains report parameters, making it easy to see the options (mainly time range and filters today) that are driving the report being viewed

genTimeRangeSelectorAndControlForm

File

`vovcgi.tcl`

Usage

```
genTimeRangeSelectorAndControlForm title sections  
url insertTopSubmit showWorkweek showFilterNegation  
showTimeRange args
```

Description

Monitor report options box, with time range selection, Ajax-driven cascading filters, and many other input types

genUrlArgs

File

`vovcgi.tcl`

Usage

```
genUrlArgs vars bCustomAccounts bPassTableFilters  
rOptions rFilters
```


Description Processes opt array, identifying the elements that should be passed on to pages that are called from within the current page (filterOptions)

genWidgetAjax

File vovcgi.tcl

Usage genWidgetAjax page widget msg options

Description Inserts a widget Ajax call into a page

GEO

File vovfdl.tcl

Usage GEO args

Description Set the geometry (placement) of an object in the context of a set. Two syntaxes are supported:

- GEO setId objectId x y width height
- GEO x y width height

In the second format, the `setId` and `objectId` are automatically assigned to the most recent set and object in FDL, so this syntax is generally best to use immediately following a set, job, or file (input/output). If the most recent object is a set, the parent set of that set is used for the context set (`setId`). For the second syntax, `make(lastId)` and `make(setIdStack)` are used to determine the correct IDs to use.

getCommonOptionsQueryString

File vovcgi.tcl

Usage getCommonOptionsQueryString excludeList (default=)

Description Returns all options found in the current URL but does not generate hidden inputs. Also converts spaces to + for Tcl security.

getComponentOption

File vovcgi.tcl

Usage getComponentOption comp opt rtn (default=)

Description Gets a specific option for an existing component

getCssName

File vovcgi.tcl

Usage getCssName inputName ext (default=css)

getData

File vovregistrylib.tcl

Usage GetData

getDataFrom

File vovregistrylib.tcl

Usage GetDataFrom filePathFull

getEmailAddress

File vovnotifylib.tcl

Usage getEmailAddress user

Description This procedure can be overridden in the vovnotifyd/config.tcl file.

getHost

File vovcgi.tcl

Usage getHost

GetHostName

File vovregistrylib.tcl

Usage GetHostName

GetJsonFormatted

File vovregistrylib.tcl

Usage GetJsonFormatted dictInput

getJSScriptName

File vovcgi.tcl

Usage getJSScriptName inputName ext (default=js)

getMailToProp

File vovnotifylib.tcl

Usage getMailToProp jobid now

getMailWhatProp

File vovnotifylib.tcl

Usage getMailWhatProp jobid now

getMailWhatText

File vovnotifylib.tcl

Usage getMailWhatText jobid maxlen (default=0)

Description Get the logfile text for jobid to mail for LSF emulation The email contains a hypertext link to the job.

maxlen parameter: If zero (default), get only the first 2¹⁶ bytes, If positive, get first maxlen bytes A negative value for maxlen means do not truncate.

getMenuOption

File vovcgi.tcl

Usage getMenuOption tab menu opt rtn (default=)

Description Gets a specific option for an existing menu.

getMinOrFullName

File vovcgi.tcl

Usage getMinOrFullName inputName ext

getPageOption

File vovcgi.tcl

Usage getPageOption tab menu page opt rtn (default=)

Description Gets a specific option for an existing page.

getPort

File vovcgi.tcl

Usage getPort allowReadOnlyPort

GetPortNumber

File vovregistrylib.tcl

Usage GetPortNumber configString

GetQueue

File vovregistrylib.tcl

Usage GetQueue

getReadOnlyPort

File vovcgi.tcl

Usage getReadOnlyPort

GetRegDir

File vovregistrylib.tcl

Usage GetRegDir

GetRegFile

File vovregistrylib.tcl

Usage GetRegFile

getReportUrlHostPort

File vovcgi.tcl

Usage getReportUrlHostPort allowReadOnlyPort (default=true)

getTaskerColorFromStatus

File vovcgi.tcl

Usage getTaskerColorFromStatus status

GetSubDir

File vovregistrylib.tcl

Usage GetSubDir

getTabOption

File vovcgi.tcl

Usage getTabOption tab opt rtn (default=)

Description Gets a specific option for an existing tab.

getTimeRangeFromCookieAndOptions

File vovcgi.tcl

Usage getTimeRangeFromCookieAndOptions

Description This procedure is used to support needing a cookie to remember the time selection. First we get the cookie. Then we look at the timerange option (which could be a symbolic interval or the keyword "Custom").

getUrlHostPort

File vovcgi.tcl

Usage getUrlHostPort allowReadOnlyPort (default=true)

Description Returns the URL for the product instance Uses the VOV_HOST_HTTP_NAME setting if set, VOV_HOST_NAME otherwise Uses the readonly port if available, regular port otherwise.

getUrlPath

File vovcgi.tcl

Usage getUrlPath args

GUI_LABEL

File vovfdl.tcl

Usage GUI_LABEL value

Description Set the GUI_LABEL property on the most recent node.

H

HR

File vovhtml.tcl

Usage HR args

HREF

File vovhtml.tcl

Usage HREF href text target (default=) class (default=) rel
(default=) args

HREFHINT

File vovhtml.tcl

Usage HREFHINT msg

HREFIMG

File vovhtml.tcl

Usage HREFIMG href img title args

HREFPATH

File vovhtml.tcl

Usage HREFPATH url label

HREFTOP

File vovhtml.tcl

Usage HREFTOP href text

Description Same as above, except the target is the "TOP" level window.

HTML

File vovhtml.tcl

Usage HTML script

html_encode

File vovhtml.tcl

Usage html_encode s

I

I

File vovfdl.tcl

Usage I args

Description Declare a list of inputs for the most recently declared transition `(make(transId))`. The inputs are elements in the database `make(db)`, which is normally 'FILE'. Takes as a variable number of arguments. The behavior is controlled by `make(ctrl,action)`, which can be either 'trace' or 'script'.

The flags `-db NAME`, `-links (ignored)`, `-noop (ignored)`, `-normal (ignored)`, `-trigger`, `run-trigger`, `stop-log-quote`, `-sticky`, and `-waitforfile` can also be specified. Arguments can be lists.

Returns The list of VovIds for the inputs.

Side Effects `make(lastNodeId)` is set if `make(ctrl,action)` is 'trace'.

I1

File `vovfdl.tcl`

Usage `I1 args`

Description Declares input FILES for the next job. This is faster than calling I after the job. It is limited to inputs in the FILE database.

IFJOB

File `vovfdl.tcl`

Usage `IFJOB args`

Description Add a decision node to be called after a job has completed. By default, the job is the last one defined with J or T, whose id is stored in the variable `make(transId)`

Options:

- `-label LABEL`
- `-jobId jobId`

IMG

File `vovhtml.tcl`

Usage `IMG img alt (default=) title (default=) args`

indir

File `vovutils.tcl`

Usage `indir args`

initFt

File	<code>vovcgi.tcl</code>
Usage	<code>initFt urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for FlowTracer

initLa

File	<code>vovcgi.tcl</code>
Usage	<code>initLa urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for Allocator

initLm

File	<code>vovcgi.tcl</code>
Usage	<code>initLm urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for Monitor

initNc

File	<code>vovcgi.tcl</code>
Usage	<code>initNc urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for Accelerator

initNew

File	<code>vovcgi.tcl</code>
Usage	<code>initNew</code>
Description	Initialize namespace, bring in VovHTML, initialize CGI vars, and populate the opt array with URL parameters and values

initRm

File	<code>vovcgi.tcl</code>
Usage	<code>initRm urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for ResourceMonitor

initVars

File	<code>vovcgi.tcl</code>
Usage	<code>initVars</code>
Description	Initialize CGI environment variables if needed

initWa

File	<code>vovcgi.tcl</code>
Usage	<code>initWa urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for WorkloadAnalyzer

initWx

File	<code>vovcgi.tcl</code>
Usage	<code>initWx urlOptions (default=)</code>
Description	Initialize web UI, including navigation and product-specific behavior for Accelerator Plus

inProject

File	<code>vovutils.tcl</code>
Usage	<code>inProject site script</code>
Description	Execute the given script in the context of a specified project (i.e. the given queue).

INPUT

File	<code>vovhtml.tcl</code>
Usage	<code>INPUT type name value args</code>
Description	This belongs inside of forms. Sample usage: INPUT hidden action retrace

INSTRUMENTED

File	<code>vovfdl.tcl</code>
Usage	<code>INSTRUMENTED args</code>
Description	ARGS: command line of a fully instrumented script which complies with the value of VOV_FDL_ONLY.

Return The jobId of the new job

Example

```
INSTRUMENTED ./my_cool_instrumented_script.csh a b c
```

J

J

File vovfdl.tcl

Usage J args

Description Similar to T, with the addition that the appropriate capsule is used to add inputs and outputs.

See the documentation for T to see how arguments are to be passed.

Examples

```
J vw cp $aa $bb  
J "vw cp $aa $bb"  
eval J "vw cat $listOfFiles > someFile"
```

J_FINAL

File vovfdl.tcl

Usage J_FINAL

Description

JAVASCRIPT

File vovhtml.tcl

Usage JAVASCRIPT text

JC

File vovfdl.tcl

Usage JC name script

JOBCLASS

File vovfdl.tcl

Usage JOBCLASS name script

JQ_BUTTON

File vovhtml.tcl

Usage JQ_BUTTON id label options (default=)

JS_BUTTON

File vovhtml.tcl

Usage JS_BUTTON function label options (default=)

L-O

L

File vovfdl.tcl

Usage L legalExit

Description Set the list of legal exit values for the transitions to follow.

Returns The previous setting of legal exit code.

Side Effects make (legalExit) is set.

lappend_no_dup

File vovutils.tcl

Usage lappend_no_dup list_by_reference args

ldelete

File vovutils.tcl

Usage ldelete list_by_reference element

lprepend_no_dup

File vovutils.tcl

Usage lprepend_no_dup list_by_reference args

mapHostName

File vovutils.tcl

Usage mapHostName host defaultName (default=) context (default=)

mapHostNameInit

File vovutils.tcl

Usage mapHostNameInit context (default=)

MENUHREF

File vovhtml.tcl

Usage MENUHREF link label showcolor (default=1)

N

File vovfdl.tcl

Usage N jobname predictname (default=)

Description Set the name of the subsequent jobs. The name is silently truncated to 256 chars.

N2

File vovfdl.tcl

Usage N2 jobname

Description Like N, but apply to previous job. See also R2. Mostly used with instrumented jobs and vovexport utility.

NBSP

File vovhtml.tcl

Usage NBSP

nextOptValue

File vovutils.tcl

Usage nextOptValue defaultVal (default=1) argList
(default=__Top_Level_Argv)

nt_add_default_suffix

File vovutils.tcl

Usage nt_add_default_suffix name sfx

numToOrd

File vovutils.tcl

Usage

`numToOrd num`

Description

English ordinal numbers are a little weird, thanks to the non-standard naming of 11, 12, and 13 (they are different than 1, 2, 3, 21, 22, 23, etc.). Note that 111, 211, 311, etc share the weirdness.

O

File

`vovfdl.tcl`

Usage

Description

Declare a list of outputs for the most recently declared transition (whose id is available in `make(transId)`). The outputs are elements in the database `make(db)`, which is normally 'FILE'. Takes as a variable number of arguments. The behavior is controlled by `make(ctrl,action)`, which can be either 'trace' or 'script'. Arguments can be lists. The flags -normal, -shared, -stop_barrier, -propagate_barrier, -force, -optional, -rcpc, -sticky, -log, -ignore_timestamp, -noforce and -db NAME can also be specified. The flag -noop is simply ignored, and is used for the scripts where it is necessary to provide at least one flag. The flag -links is also ignored in FDL. To avoid building of inconsistent flows, the declaration is ignored if the flag '-optional' is specified.

Returns

The list of VovIds for the outputs.

Side Effects

`make(lastNodeId)` is set if `make(ctrl,action)` is 'trace'.

O1

File

`vovfdl.tcl`

Usage

`O1 args`

Description

Declares output FILES for the next job. This is faster than calling `o` after the job. It is limited to outputs in the FILE database.

O_CONFLICT

File

`vovfdl.tcl`

Usage

`O_CONFLICT handlerName`

Description

This procedure decides how to handle output conflicts that can be caused by calling the procedure `O`.

Usage: `O_CONFLICT ABORT|RETRY|name_of_handler`

This procedure sets the handler for output conflicts in FDL. The default behavior is to ABORT. It is also possible to choose RETRY, which kills the currently running old job and then retries the output declaration. It is also possible to pass the name of a custom handler. This handler takes 3 arguments: jobId outputId oldJobId and must return \"RETRY\" if successful.

OPTION

File vovhtml.tcl

Usage OPTION value selected (default=0) display (default=) args

OUT

File vovhtml.tcl

Usage OUT text

OUTLN

File vovhtml.tcl

Usage OUTLN text

P

P

File vovfdl.tcl

Usage P propertyname propertyvalue args (default=)

Description Set a sticky property on the most recent object, identified by make(lastNodeId)

page

File vovcgi.tcl

Usage page title curtab curmenu curpage (default=) body_script (default=) docUrl (default=/doc/bookshelf/index.htm) urlOptions (default=)

Description Create a product-branded page, with banner, tabbed-based navigation, and footer

PARALLEL

File vovfdl.tcl

Usage PARALLEL script

Description All of the tasks, jobs, and sets defined in the script are set up to be executed in parallel. Artificial dependencies are created between each task, job, or all the jobs in the set to build the graph in parallel. Can be nested in any order with sets, jobs, tasks, other SERIAL scripts, and other PARALLEL scripts.

Example

```
PARALLEL {  
  SERIAL {  
    TASK date 1920  
    TASK date 1921  
    TASK date 1922  
  }  
  PARALLEL {  
    SERIAL {  
      TASK date 1923  
      TASK date 1924  
      TASK date 1925  
    }  
    SERIAL {  
      TASK date 1926  
      TASK date 1927  
      TASK date 1928  
    }  
  }  
}
```

PARRAY

File vovcgi.tcl

Usage PARRAY a prefix (default=)

Description Print a Tcl array in preformatted text

parseQueryString

File vovcgi.tcl

Usage parseQueryString

Description Populate the opt array with URL parameters and values

PJ

File vovfdl.tcl

Usage PJ args

Description

Define a periodic job.

Old syntax: The first 3 arguments define the target, min, and max period.

```
Syntax:
-period <TIMESPEC> -- Target period of job
-min <TIMESPEC> -- Minimum period of job
-max <TIMESPEC> -- Maximum period of job
-autokill <TIMESPEC> -- Kill job if it runs longer
than this
-resources <RESOURCES> -- Override resources
-r <RESOURCES> -- Same as -resources
-env <ENVIRONMENT> -- Specify named environment for
job
-e <ENVIRONMENT> -- same as -env
-systemjob -- The job is considered "systemjob"
(i.e. not considered for server auto-shutdown)
-cal <CALENDARSPEC> -- Add an additional calendar
constraint
Examples: "*:23" "Sun:*" "Sat,Sun:2,3,4"
-P <TIMESPEC> -- Set -period, -min, -max, and -
autokill values
all at once with: min = p, max=10*p and autokill=3*p.
```

Remaining arguments are passed to the FDL `J` procedure. The job is attached to the set `System:vovperiodicd`. The properties `PERIOD` `PERIOD_MIN` and `PERIOD_MAX` are set on the job.

If `-cal` is given with a calendar spec, the property `PERIOD_CAL` is set on the job.

If property `PERIOD_PAUSE` on the job is set to 1, the job is paused. If this is set while the job is running, the job continues.

Returns

The VovId of the transition.

Side Effects

`make(lastNodeId)` and `make(transId)` are set.

prettyPrintBytes

File

`vovutils.tcl`

Usage

`prettyPrintBytes b`

printenv

File

`vovenutils.tcl`

Usage

`printenv`

Description

Print the global array `env`.

PRIORITY

File	vovfdl.tcl
Usage	PRIORITY spri xpri (default=)
Description	Define priority for subsequent jobs. The first argument is the scheduling priority. The second, optional, is the execution priority. The priority will be used only if you retrace at the "same" priority. else the priority is likely to be overridden by the retrace request.

processAccessRule

File	vovcgi.tcl
Usage	processAccessRule rule parentArray
Description	Processes an access rule, keeping the role separate from users/groups

puts

File	vovutils.tcl
Usage	puts args

R

R

File	vovfdl.tcl
Usage	R reslist script (default=)
Description	Set default resources for the jobs to follow (R must come before T or J).
Returns	Nothing
Side Effects	With one argument, make(resources) is set. If the script argument is also given, the procedure restore the current value of make(resources)

R2

File	vovfdl.tcl
Usage	R2 reslist

Description

Similar to `R`, only it applies only to the previous job
`make (transId)`. See also `N2`.

RAWINPUT

File

`vovhtml.tcl`

Usage

`RAWINPUT type name value args`

Description

This belongs inside of forms. Sample usage: `INPUT hidden action
retrace`

RAWOUT

File

`vovhtml.tcl`

Usage

`RAWOUT text`

Description

Add 'text' to the HTML page being built

RAWOUTLN

File

`vovhtml.tcl`

Usage

`RAWOUTLN text`

RCPC

File

`vovfdl.tcl`

Usage

`RCPC args`

Examples

```
RCPC [glob *.o]
RCPC -off [glob *.o]
RCPC 00123456 00234567
```

readWebCfg

File

`vovcgi.tcl`

Usage

`readWebCfg`

Description

Read the web configuration file to obtain user customizations.

RECONCILE_WITH_FILE_SYSTEM

File

`vovtraceutils.tcl`

Usage

`RECONCILE_WITH_FILE_SYSTEM args`

RUNMODE

File

Usage

S

S+

File

vovfdl.tcl

Usage

S+ setname args

Description

Simpler version of s.

- S+ setname -last: --- Add last node to setname
- S+ setname -id: id1
- S+ setname -idlist "id1 id2": Same as -id
- S+ setname -set: setname2
- S+ setname script: Add jobs in script to setname

S

File

vovfdl.tcl

Usage

S setname script args

Description

All the jobs defined in script go into the set setname. If setname begins with a + or if setname does not have a colon (:) then the setname is considered a leaf and it is appended to the setname of the containing "S" (using : as separator).

Options

- -gui_label: label for the GUI
- -files: Also include files into the set

Returns

the id of the set 'setname'

Examples

```
S myset {  
  J vw cp aa bb  
}  
  
S myset {  
  S s1 {  
    J vw cp aa bb  
  }  
  S s2 {  
    J vw cp bb cc  
  }  
}
```

```
} -gui_label "Hi there.\nThis is a set with 2
subsets"
```

S_update_files

File

vovfdl.tcl

Usage

S_update_files setname

Description

All the jobs defined in script go into the set setname. If setname begins with a + or if setname does not have a colon (:) then the setname is considered a leaf and it is appended to the setname of the containing "S"

Options

- -gui_label: label for the GUI
- -files: Also include files into the set

Returns

The ID of the set 'setname'

Examples

```
S myset {
  J vw cp aa bb
}

S myset {
  S s1 {
    J vw cp aa bb
  }
  S s2 {
    J vw cp bb cc
  }
} -gui_label "Hi there.\nThis is a set with 2
subsets"
```

S_update_set

File

vovfdl.tcl

Usage

S_update_files setname

Description

Adds files to the contents of the named set. By default, the " S " proc only adds JOBS to the content of the set.

safeshift

File

vovutils.tcl

Usage

safeshift argList (default=__Top_Level_Argv)

SCRIPT

File vovhtml.tcl

Usage SCRIPT text

SELECT_ALT

File vovhtml.tcl

Usage SELECT_ALT size name script

SELECT_MULTIPLE

File vovhtml.tcl

Usage SELECT_MULTIPLE size name script

sendMail

File vovnotifydlib.tcl

Usage sendMail recipients subject body

Description This procedure can be over-ridden in the vovnotifyd/config.tcl file. Send an email with subject and body to each address in recipients. Has ability to use external program specified in NOTIFYD(mailprog). If the above variable is unset or contains the value 'SMTP' the original behavior of using direct SMTP with the parameters specified in the NOTIFYD array is obtained. The mailprog may be a full path (/bin/mail) or leafname (mailx). If a leafname, the program executable is located with vtk_which. The mailprog must accept the body on stdin, use -s for subject.

sendMailInit

File vovnotifydlib.tcl

Usage sendMailInit

SERIAL

File vovfld.tcl

Usage SERIAL script

Description All of the tasks, jobs, and sets defined in the script are set up to be executed serially in the order they appear. Artificial dependencies are created between each task, job, or all the jobs in the set to build the graph serially. Can be nested in any order

with sets, jobs, tasks, other SERIAL scripts, and other PARALLEL scripts.

Example:

```
SERIAL {
  TASK make clean
  TASK configure
  TASK make
  TASK make install
}

SERIAL {
  S Run_Three_A_In_Parallel {
    PARALLEL {
      T vw scriptA1.csh
      T vw scriptA2.csh
      T vw scriptA3.csh
    }
  }
  S Run_Three_B_In_Parallel {
    PARALLEL {
      T vw scriptB1.csh
      T vw scriptB2.csh
      T vw scriptB3.csh
    }
  }
}
```

SERVER_PARAMETER

File

vovtraceutils.tcl

Usage

SERVER_PARAMETER name value

Description

This procedure was used mostly with the output from vovshow - policy

setComponentOption

File

vovcgi.tcl

Usage

setComponentOption comp opt val

Description

Sets a specific option for an existing component.

setenv

File

vovenvutils.tcl

Usage

setenv var value

Description

Set the environment variable to the given value.

Return

value

Side Effects	both env(VAR) and VAR are set.
setFooter	
File	vovcgi.tcl
Usage	setFooter html
Description	Sets a custom footer to be shown above the Altair Engineering copyright statement.
setJsDebug	
File	vovcgi.tcl
Usage	setJsDebug
setMenu	
File	vovcgi.tcl
Usage	setMenu tab menu
Description	Sets active tab and menu when using VovCGI::addPage
setMenuOption	
File	vovcgi.tcl
Usage	setMenuOption tab menu opt val
Description	Sets a specific option for an existing menu
setOpt	
File	vovcgi.tcl
Usage	setOpt n v
Description	Set an element in the opt array if it does not already exist Used to set defaults after the VovCGI namespace has been initialized.
setPageMessage	
File	vovhtml.tcl
Usage	setPageMessage msglevel msg
setPageOption	
File	vovcgi.tcl

Usage `setPageOption tab menu page opt val`

Description Sets a specific option for an existing page

setTab

File `vovcgi.tcl`

Usage `setTab tab`

Description Sets active tab when using VovCGI::addMenu

setTabOption

File `vovcgi.tcl`

Usage `setTabOption tab opt val`

Description Sets a specific option for an existing tab.

setUrlOptions

File `vovcgi.tcl`

Usage `setUrlOptions args`

setUrlPath

File `vovcgi.tcl`

Usage `setUrlPath ur`

SetUseFileFromRegistry

File `vovregistrylib.tcl`

Usage `SetUseFileFromRegistry state`

setValidUrlOptions

File `vovcgi.tcl`

Usage `setValidUrlOptions list`

shift

File `vovutils.tcl`

Usage `shift argList (default=__Top_Level_Argv`

showDeniedFileAccessMessage

File	vovcgi.tcl
Usage	showDeniedFileAccessMessage
Description	Print message explaining that a certain page or page element is not accessible by the authenticated user

showFileAsImage

File	vovcgi.tcl
Usage	showFileAsImage id file args

showFilterOptions

File	vovcgi.tcl
Usage	showFilterOptions bNegate filters
Description	Currently used by Im_batch.mod only

showHideOptions

File	vovcgi.tcl
Usage	showHideOptions
Description	Helper procedure used to show or hide the options box above

showMessages

File	vovcgi.tcl
Usage	showMessages productName title curtab curmenu

showPageMessage

File	vovhtml.tcl
Usage	showPageMessage

SMTP

File	vovnotifydlib.tcl
Usage	SMTP sock msg

SourceHostMapFile

File	vovutils.tcl
-------------	--------------

Usage `SourceHostMapFile fn`

START

File `vovfdl.tcl`

Usage `START args`

Description A convenient way to start a retrace in FDL. The argument list consists of one or more VovIds for objects. Each objectId can refer to either a node or a set. The retracing is started in SAFE mode at NORMAL priority. For more control, please use `vtk_retrace_start` directly.

stdOutFailover

File `vovutils.tcl`

Usage `stdOutFailover args`

STOP

File `vovfdl.tcl`

Usage `STOP args`

T

T_FINAL

File `vovfdl.tcl`

Usage `T args`

Description Declare a transition in the current working directory and in the current environment stored in `make(env)`.

The transition's resources are determined by `make(resources)`.

The variable `make(runmode)` may be set to control how to execute the job (`normal`, `xterm_icon`, `xterm_open`, `xterm_none`). The runmode can also be overridden with the options `-xb -xi -xo -xn`.

If the transition already exists, it is left unchanged.

If the command line is complex, you should quote the entire command in a single argument, as shown in the second example below.

Examples:

```
T vw cp aa bb
T "vw cat $listOfFiles > someFile"
T -xi "vw cat $listOfFiles > someFile"
eval T vw cat $listOfFiles > someFile
```

T_FINAL

File

vovfdl.tcl

Usage

T_FINAL

tag

File

vovhtml.tcl

Usage

tag args

TASK

File

vovfdl.tcl

Usage

TASK args

Description

Declare a task. This is a wrapper for the procedure T, to be used within a SERIAL or PARALLEL script.

Example:

```
TASK make clean
TASK make install
```

Returns

The VovId of the job.

TEXTAREA

File

vovhtml.tcl

Usage

TEXTAREA name cols rows content args

TimeRangeSelector

File

vovcgi.tcl

Usage

TimeRangeSelector now showCurrentDay

Description

Inserts a time range selector, with both predefined selections and a custom time range input that uses a JQuery-UI based calendar input widget

TIMEVAR

File	vovtimevar.tcl
Usage	TIMEVAR label list
Description	Define rules to be computed at regular intervals. Typically used in the initialization of job classes (<code>initJobClass</code> procedure). This sets the variable <code>TIMEVAR(list)</code> .

TIMEVAR_evaluate

File	vovtimevar.tcl
Usage	TIMEVAR_evaluate now fromFiles (default=false)
Description	This is the procedure that evaluates all rules defined by the TIMEVAR procedures, which is kept in the variable <code>TIMEVAR(list)</code> . This procedure is called, typically, once a minute.

TIMEVAR_evaluate_now

File	vovtimevar.tcl
Usage	TIMEVAR_evaluate_now fromFiles (default=false)

TITLE

File	vovhtml.tcl
Usage	TITLE title

tw_begin

File	vov_twheels.tcl
Usage	tw_begin label

tw_button

File	vov_twheels.tcl
Usage	tw_button label var helpMsg (default=)

tw_command

File	vov_twheels.tcl
Usage	tw_command label command helpMsg (default=)

tw_end

File vov_twheels.tcl

Usage tw_end

tw_variable

File vov_twheels.tcl

Usage tw_variable label var deflt bindCmd (default=)

U

unsetenv

File vovenvutils.tcl

Usage unsetenv args

Description Unset the environment variables given as arguments.

Side Effects Both env(VAR) and VAR are unset.

unshift

File vovenvutils.tcl

Usage unshift arg argList (default=__Top_Level_Argv)

UpdateValue

File vovregistrylib.tcl

Usage UpdateValue key value

UpdateValues

File vovregistrylib.tcl

Usage UpdateValues dictNew

url_decode

File vovhtml.tcl

Usage url_decode s

url_encode

File vovhtml.tcl

Usage `url_encode string`

Description do x-www-urlencoded character mapping.

The spec says: "non-alphanumeric characters are replaced by '%HH'"

1. Leave alphanumerics characters alone
2. Convert every other character to an array lookup
3. Escape constructs that are "special" to the Tcl parser
4. "subst" the result, doing all the array substitutions

urlDecode

File `vovhtml.tcl`

Usage `urlDecode s`

Description This procedure retains the representation of spaces as + symbols

UseFileFromRegistry

File `vovregistrylib.tcl`

Usage `UseFileFromRegistry`

USERTXT

File `vovutils.tcl`

Usage `USERTXT code msg args`

V

V

File `vovfdl.tcl`

Usage `V listOfTests`

Description V stands for verify. The input parameter *listOfTests* is a space-separated list of tests to be performed by the program `CheckTest`. See the documentation for `CheckTest` for more information on the syntax of these tests.

For test, i.e. for each element in the input list, this procedure adds an annotation whose text is `TEST=$test`, to the most recently defined transition, stored in `make(transId)`.

Side Effects	All annotations with text beginning with TEST= are replaced.
ves	
File	vovenvutils.tcl
Usage	ves environment
Description	Similar to the shell alias to switch environment.
VFmakeProperSetName	
File	vovfdl.tcl
Usage	VFmakeProperSetName setname
VncFmtTaskerReservation	
File	vovutils.tcl
Usage	VncFmtTaskerReservation taskername
VncGetNcConfigDir	
File	vovutils.tcl
Usage	VncGetNcConfigDir
VncJobClassSearchPath	
File	vovutils.tcl
Usage	VncJobClassSearchPath
Description	<p>Modeled on <code>vovenv_dirs</code>; in <code>vovutils</code> so accesible to <code>vovresourced</code></p> <p>Uses the variable <code>env(VOV_JOBCLASS_DIRS)</code>. On UNIX, this is a colon separated list of directories. On Windows, the separator is semicolon.</p>
Returns	List of directories in which jobclass files can be found. First, dirs named by <code>env(VOV_JOBCLASS_DIRS)</code> , then <code>\$VOVDIR/local/jobclass</code> and finally <code>\$VOVDIR/etc/jobclass</code> .
VncSourcePolicyFiles	
File	vovutils.tcl
Usage	VncSourcePolicyFiles
Description	Now source the project or site-specific policy files.

VncSourceQueueConfigFile

File	vovutils.tcl
Usage	VncSourceQueueConfigFile qnameOrSetupFile quiet (default=0)
Description	Set the values of VOV_HOST_NAME and VOV_PROJECT_NAME to locate the Accelerator server, and open the trace.

vov_gcd

File	vov_common.tcl
Usage	vov_gcd u v

vov_getImage

File	vov_common.tcl
Usage	vov_getImage imageName

vov_getImageCopy

File	vov_common.tcl
Usage	vov_getImageCopy imgName

vov_makeImage

File	vov_common.tcl
Usage	vov_makeImage imageName

vov_resizeImage

File	vov_common.tcl
Usage	vov_resizeImage img num denom background_color (default=)

vov_ssd_uniq_port

File	vov_un2id.tcl
Usage	vov_ssd_uniq_port user (default=)

vov_un2id

File	vov_un2id.tcl
Usage	vov_un2id s max (default=0xffff) min (default=0)

Description	Convert a string to a positive integer within the specified range (default 0 - 0xffff)
vov_update	
File	vov_common.tcl
Usage	vov_update why idle (default=0)
vov_update_idle	
File	vov_common.tcl
Usage	vov_update why idle
vov_update_idletasks	
File	vov_common.tcl
Usage	vov_update_idletasks why
vovAddPhantomInputs	
File	vovphantom.tcl
Usage	vovAddPhantomInputs jobId searchPath n (default=1)
Description	This procedure adds PHANTOM inputs to a job, based on a search path. The third parameter is the number of components in the path that are to be used as tail.
VovBarrierClose	
File	vovutils.tcl
Usage	VovBarrierClose fp
VovBarrierOpen	
File	vovutils.tcl
Usage	VovBarrierOpen file mode (default=664)
VovChmod	
File	vovutils.tcl
Usage	VovChmod pathName newPermissions

vovClientInit

File	<code>vov_boot.tcl</code>
Usage	<code>vovClientInit flag</code>
Description	Initialize the TCL interface. The argument flag is passed to <code>vtk_init</code> and can be one of "tcl" "tk" "tix" "all".
Returns	Either "ok" or raises an error if initialization fails.
Side Effects	calls <code>{\tt vovEnableTk}</code>

VovCompareTimeSpec

File	<code>vovutils.tcl</code>
Usage	<code>VovCompareTimeSpec t1 t2</code>

vovConflictControl

File	<code>vov_messages.tcl</code>
Usage	<code>vovConflictControl</code>
Description	Check the value of the <code>VOV_CONFLICT_CONTROL</code> . If it is <code>ABORT</code> or <code>CONTINUE</code> , return its value. If it has another value, print a warning and return <code>ABORT</code> . If it is not set, return the empty string. For backward compatibility, check also <code>VOV_FAIL_ON_CONFLICT</code> : if it is set, return <code>ABORT</code> .

vovCreateLogDirUltraSafe

File	<code>vovenvutils.tcl</code>
Usage	<code>vovCreateLogDirUltraSafe logdir args</code>
Description	This one is hard to test. At Samsung, the file <code>mkdir \$dir</code> command fails with the error "Can't create directory: file already exists" So, if we get that error, we just try again, slowly.

VovDaemon::checkConfigFile

File	<code>vovdaemonlib.tcl</code>
Usage	<code>VovDaemon::checkConfigFile daemonName</code>

VovDaemon::checkInfoFile

File	<code>vovdaemonlib.tcl</code>
Usage	<code>VovDaemon::checkInfoFile daemon args</code>

VovDaemon::initVariables

File	vovdaemonlib.tcl
Usage	VovDaemon::initVariables daemonName

VovDaemon::loadFile

File	vovdaemonlib.tcl
Usage	VovDaemon::loadFile file

VovDaemonMgrReread

File	vovdaemonlib.tcl
Usage	VovDaemonMgrReread daemon

vovDestroyTopLevel

File	vov_common.tcl
Usage	vovDestroyTopLevel w (default=)
Description	Destroy the top level. If the toplevel is the main application window it will be destroyed and the application will exit.

vovDialog

File	vov_dialog.tcl
Usage	vovDialog title text bitmap default args
Description	This is a minimal modification of the default {\tt dialog} normally distributed with Tk. This procedure displays a dialog box, waits for a button in the dialog to be invoked, then returns the index of the selected button.

VovDiskFullCallback

File	vov_boot.tcl
Usage	VovDiskFullCallback code

vovDroppedAsyncConnectionCallBack

File	vov_boot.tcl
Usage	vovDroppedAsyncConnectionCallBack

VovDumpPreemptionRules

File vovtraceutils.tcl

Usage VovDumpPreemptionRules fileName

VovEdition

File vovutils.tcl

Usage VovEdition

vovEnableTk

File vov_boot.tcl

Usage vovEnableTk

Description Initialize the part of the Tcl interface that has to do with Tk.

Side Effects Colors are initialized

vovenv

File vovenvutils.tcl

Usage vovenv var separator op args

Description A utility to manage the environment variables, such as PATH or LM_LICENSE_FILE, whose value is a list of strings separated by a special character.

- var is the name of the variable,
- separator is ":" in UNIX and ";" in Windows NT or "auto" (recommended) to automatically choose between ";" and ":".
- op is one of ADD APPEND PREPEND DELETE,
- The remaining args are the components that must be added or removed from the variable. In all cases, duplicates will be avoided.

Return The new value of the environment variable.

Side Effects The value of the variable is modified with setenv.

vovenv_current_env

File vovenvutils.tcl

Usage vovenv_current_env

Return The name of the current environment.

Side Effects env(VOV_ENV) is set if it is not defined.

vovenv_do_switch

File vovenvutils.tcl

Usage vovenv_do_switch new_env_spec

Description The new environment specification is a list of environment names, each with optional parameters, separated by + and - signs. The optional parameters are shown in parenthesis and are separated by commas.

Examples: E1 E1+E2-E3 +E2-E3 E1(p1,p2)+E2+E3(p1)

If the new environment specification begins with a '+', the environment is added to the current one, else it replaces it.

Return "success" or "failure"

vovenv_produce_script

File vovenvutils.tcl

Usage vovenv_produce_script shelltype file

Description Produce a script that can be sourced by the shell specified by shelltype The script only contains the variables that have changed.

Return "success" or "failure"

vovenvCaptureEnvironment

File vovenvutils.tcl

Usage vovenvCaptureEnvironment logdir force addToFlow
(default=0)

Description Capture the environment in a file, if necessary.

Return envfile

Side Effects Set env(VOV_ENV) -- If we are in a legitimate environment, use it by default. -- Else use an environment file.

vovenvCreateSnapshotPropertyValue

File vovenvutils.tcl

Usage vovenvCreateSnapshotPropertyValue

vovenvReadSnapshotPropertyFromObject

File	<code>vovenvutils.tcl</code>
Usage	<code>vovenvReadSnapshotPropertyFromObject objId</code>

vovenvSaveSnapshotPropertyOnObject

File	<code>vovenvutils.tcl</code>
Usage	<code>vovenvSaveSnapshotPropertyOnObject objId</code>

VovError

File	<code>vovutils.tcl</code>
Usage	<code>VovError msg</code>

VovErrorCount

File	<code>vovutils.tcl</code>
Usage	<code>VovErrorCount</code>

VovExit

File	<code>vovutils.tcl</code>
Usage	<code>VovExit status</code>

VovExpandBang

File	<code>vovutils.tcl</code>
Usage	<code>VovExpandBang</code>
Description	The ! ("bang") is expanded as the most recent my job in the current directory.

VovFatalError

File	<code>vovcgi.tcl</code>
Usage	<code>VovFatalError msg args</code>

VovFeedback

File	<code>vovutils.tcl</code>
Usage	<code>VovFeedback msg</code>

vovFindGoodEnvFile

File vovenvutils.tcl

Usage vovFindGoodEnvFile envfile

VovForgetIdleLimits

File vovtraceutils.tcl

Usage VovForgetIdleLimits minAge (default=0)

vovForgetIsolatedFiles

File vovtraceutils.tcl

Usage vovForgetIsolatedFiles verbose (default=1) update (default=1)

Description Forget all files that are isolated. If verbose is set, print a message for each file that is forgotten. If update is set, the Tk function update is called after each node has been processed. We do not touch isolated nodes that appear to have just been created (the node timestamp is recent).

Return The number of nodes that have been forgotten.

vovForgetIsolatedNodes

File vovtraceutils.tcl

Usage vovForgetIsolatedNodes verbose (default=1) update (default=1)

vovForgetNodesByRule

File vovtraceutils.tcl

Usage vovForgetNodesByRule rule

Description Forget all nodes that match the given rule.

Return The number of forgotten nodes.

VovForgetOldUnusedLimits

File vovtraceutils.tcl

Usage VovForgetOldUnusedLimits thresholdSpec (default=7d)

vovForgetRetracingBlockingNodes

File	<code>vovtraceutils.tcl</code>
Usage	<code>vovForgetRetracingBlockingNodes verbose (default=1) update (default=1)</code>
Description	Forget all files that with status RETRACING and with no inputs
Return	The number of nodes that have been forgotten.

VovGetLocalHostName

File	<code>vovutils.tcl</code>
Usage	<code>VovGetLocalHostName</code>
Description	Returns hostname as vovserver sees it.

VovGetNullDevice

File	<code>vovutils.tcl</code>
Usage	<code>VovGetNullDevice</code>

VovGetNumericPriority

File	<code>vovutils.tcl</code>
Usage	<code>VovGetNumericPriority s</code>

VovGetNumericSecurity

File	<code>vovutils.tcl</code>
Usage	<code>VovGetNumericSecurity s</code>

vovGetPortNumberFromEnv

File	<code>vovutils.tcl</code>
Usage	<code>vovGetPortNumberFromEnv envName defaultValue (default=)</code>
Description	@ args envName Typically VOV_PORT_NUMBER defaultValue, in case the variable is not defined.

VovGetPriorityChars

File	<code>vovutils.tcl</code>
Usage	<code>VovGetPriorityChars n</code>

VovGetProductName

File	<code>vovutils.tcl</code>
Usage	<code>VovGetProductName</code>

vovGetProjectFileName

File	<code>vovtraceutils.tcl</code>
Usage	<code>vovGetProjectFileName pf serverdir (default=)</code>

VovGetProp

File	<code>vovutils.tcl</code>
Usage	<code>VovGetProp objId name dflt</code>

VovGetPropertyWithTimeout

File	<code>vovutils.tcl</code>
Usage	<code>VovGetPropertyWithTimeout objId propName args</code>
Description	A robust implementation of <code>vtk_prop_get</code> with timeout, default values, etc. This implementation tries multiple times

VovGetRegistry

File	<code>vovregistrylib.tcl</code>
Usage	<code>VovGetRegistry</code>
Description	Return the path to the VOV registry directory

VovGetRegistryFiles

File	<code>vovregistrylib.tcl</code>
Usage	<code>VovGetRegistryFiles</code>

VovGetSecurityLevel

File	<code>vovutils.tcl</code>
Usage	<code>VovGetSecurityLevel</code>
Description	Return the numeric value of the current user security. 2 = READONLY 3 = USER 4 = LEADER

5 = ADMIN

vovGetSnapshotDir

File vovenvutils.tcl

Usage vovGetSnapshotDir logdir

Description Try to keep snapshots away from the logs, to avoid NFS caching.

VovGetSubsets

File vovtraceutils.tcl

Usage VovGetSubsets setIdOrName

VovGetSymbolicSecurity

File vovutils.tcl

Usage VovGetSymbolicSecurity n

VovGetToolName

File vovfdl.tcl

Usage VovGetToolName name

Description Get the characteristic name of a tool (on NT, drop the .exe .bat suffix).

VovGetUniqueJobs

File vovtraceutils.tcl

Usage VovGetUniqueJobs doRecursive doVerbose allJobsRef setId

VovGetUserName

File vovutils.tcl

Usage VovGetUserName

VovGetVovPortFromWebPort

File vovtraceutils.tcl

Usage VovGetVovPortFromWebPort webhost webport useSsl

Description set webhost [lindex [split \$queue "@"] 1] set webport \$result(\$queue,webport)

vovHashEnvironment

File	<code>vovenvutils.tcl</code>
Usage	<code>vovHashEnvironment</code>

vovHelp

File	<code>vov_common.tcl</code>
Usage	<code>vovHelp txt</code>

VovHierFlow::BLOCK

File	<code>vovhierflow.tcl</code>
Usage	<code>VovHierFlow::BLOCK name type arg1 args</code>

VovHierFlow::C

File	<code>vovhierflow.tcl</code>
Usage	<code>VovHierFlow::C args</code>

VovHierFlow::computeSubFlowStats

File	<code>vovhierflow.tcl</code>
Usage	<code>VovHierFlow::computeSubFlowStats subFlowName stats w (default=)</code>

VovHierFlow::customize

File	<code>vovhierflow.tcl</code>
Usage	<code>VovHierFlow::customize window menu</code>

VovHierFlow::DominantStatus

File	<code>vovhierflow.tcl</code>
Usage	<code>VovHierFlow::DominantStatus s1 s2</code>

VovHierFlow::FOLDER

File	<code>vovhierflow.tcl</code>
Usage	<code>VovHierFlow::FOLDER name args</code>
Description	<ul style="list-style-type: none">• <code>-doc DOCS -side SIDE</code>: Where side can be "left" "right" "top" "bottom" (as in Tk pack)

- -balloon MvSG -buttons B: Where B is one of "vertical","horizontal","none"
- -hide: Hide the folder (frame not visible, but there)
- -label L: Where L can be "normal" or "none"

VovHierFlow::GetFromParentFolder

File vovhierflow.tcl

Usage VovHierFlow::GetFromParentFolder what dflt

VovHierFlow::HLF

File vovhierflow.tcl

Usage VovHierFlow::HLF name script args

VovHierFlow::HLFAddBlockMenu

File vovhierflow.tcl

Usage VovHierFlow::HLFAddBlockMenu w setName type

VovHierFlow::HLFAddSetControls

File vovhierflow.tcl

Usage VovHierFlow::HLFAddSetControls f setname side

VovHierFlow::HLFAddSubFlowControls

File vovhierflow.tcl

Usage VovHierFlow::HLFAddSubFlowControls f subFlowName

VovHierFlow::HLFballoon

File vovhierflow.tcl

Usage VovHierFlow::HLFballoon w msg

Description \$::VovGUI::widget(balloon) bind \$w -balloonmsg \$msg

VovHierFlow::HLFbindings

File vovhierflow.tcl

Usage VovHierFlow::HLFbindings w setName

VovHierFlow::HLFdoc

File	vovhierflow.tcl
Usage	VovHierFlow::HLFdoc w docs

VovHierFlow::HLFgetContext

File	vovhierflow.tcl
Usage	VovHierFlow::HLFgetContext

VovHierFlow::HLFinitializeStructure

File	vovhierflow.tcl
Usage	VovHierFlow::HLFinitializeStructure

VovHierFlow::HLFpopContext

File	vovhierflow.tcl
Usage	VovHierFlow::HLFpopContext

VovHierFlow::HLFpopupMenu

File	vovhierflow.tcl
Usage	VovHierFlow::HLFpopupMenu name x y type w

VovHierFlow::HLFpushContext

File	vovhierflow.tcl
Usage	VovHierFlow::HLFpushContext f

VovHierFlow::HLFrender

File	vovhierflow.tcl
Usage	VovHierFlow::HLFrender name top

VovHierFlow::HLFrenderInside

File	vovhierflow.tcl
Usage	VovHierFlow::HLFrenderInside name w

VovHierFlow::HLFrenderNewWindow

File	vovhierflow.tcl
Usage	VovHierFlow::HLFrenderNewWindow name

VovHierFlow::HLFshowDocs

File vovhierflow.tcl

Usage VovHierFlow::HLFshowDocs docSpec

VovHierFlow::HLFStartVovConsole

File vovhierflow.tcl

Usage VovHierFlow::HLFStartVovConsole proj host port setName

VovHierFlow::HLFStartVovProject

File vovhierflow.tcl

Usage VovHierFlow::HLFStartVovProject proj

VovHierFlow::HLFupdateSetLabel

File vovhierflow.tcl

Usage VovHierFlow::HLFupdateSetLabel tkLabel blockName setName

VovHierFlow::HLFupdateStatusBarForSet

File vovhierflow.tcl

Usage VovHierFlow::HLFupdateStatusBarForSet tkcanvas STAT

VovHierFlow::HLFupdateSubFlow

File vovhierflow.tcl

Usage VovHierFlow::HLFupdateSubFlow label subFlowName

VovHierFlow::HLFupdateSubProject

File vovhierflow.tcl

Usage VovHierFlow::HLFupdateSubProject label name proj host
port setName

VovHierFlow::HLFUpdateViewButtonForSet

File vovhierflow.tcl

Usage VovHierFlow::HLFUpdateViewButtonForSet f action setId

VovHierFlow::HLFUpdateViewButtonForSubFlow

File vovhierflow.tcl

Usage VovHierFlow::HLFUpdateViewButtonForSubFlow f action
flowName

VovHierFlow::iconName

File vovhierflow.tcl

Usage VovHierFlow::iconName icon

VovHierFlow::init

File vovhierflow.tcl

Usage VovHierFlow::init

VovHierFlow::RunAllInSubflow

File vovhierflow.tcl

Usage VovHierFlow::RunAllInSubflow subFlowName

VovHierFlow::RunCustomize

File vovhierflow.tcl

Usage VovHierFlow::RunCustomize window menu

VovHierFlow::SETrenderInside

File vovhierflow.tcl

Usage VovHierFlow::SETrenderInside w setId args

VovHierFlow::SETrenderNewWindow

File vovhierflow.tcl

Usage VovHierFlow::SETrenderNewWindow setname

VovHierFlow::SPACER

File vovhierflow.tcl

Usage VovHierFlow::SPACER args

VovHierFlow::SUBPROJECT

File vovhierflow.tcl

Usage VovHierFlow::SUBPROJECT name projAtHost args

VOVHTML_FINISH

File	<code>vovhtml.tcl</code>
Usage	<code>VOVHTML_FINISH contentType (default=text/html)</code> <code>attachment (default=)</code>
Description	Put the completed HTML page on stdout

VOVHTML_START

File	<code>vovhtml.tcl</code>
Usage	<code>VOVHTML_START useFile (default=1) fakeDiskFull</code> <code>(default=0)</code>
Description	Start building an HTML page If useFile is nonzero, use a temporary file, else build page in memory
Returns	1 for success, 0 if errors were encountered The lines in the page are added by the proc OUT{}

vovHtmlPlotMetrics

File	<code>vovhtmlmetricplot.tcl</code>
Usage	<code>vovHtmlPlotMetrics args</code>
Description	Generate a plot of one or more metrics. with a legend on the right. Args: <ul style="list-style-type: none">• <code>-output html query raw csv</code>• <code>-title TITLE</code>• <code>-subtitle SUBTITLE</code>• <code>-resource RESOURCE</code>• <code>-width WIDTH</code>• <code>-height HEIGHT</code>• <code>-glitch SECONDS</code>• <code>-ymax MAX_Y_SHOWN_IN_PLOT</code>• <code>-bininterval TIMESPEC</code>• <code>-debug 0 1</code>• <code>-m "family metricname [color] [label]"</code> (can be repeated)

VovHttpGet

File	<code>vovutils.tcl</code>
Usage	<code>VovHttpGet host port url replyVar args</code>

Description	This procedure must be reentrant, because it calls vwait, which may cause some callbacks to be called which in turn could call this proc again. Get data from 'url' via HTTP at host:port, put reply in the variable named by replyVar. The timeout argument is in seconds and must be positive.
Returns	1 if all goes well; 0 if the socket cannot be opened or if there is a timeout.
vovInvalidateNode	
File	vovtraceutils.tcl
Usage	vovInvalidateNode id whyReason (default=)
Description	Invalidate the specified node.
Returns	1 if something was done, 0 otherwise.
VovIsCGI	
File	vov_boot.tcl
Usage	VovIsCGI
VovIsLm	
File	vovutils.tcl
Usage	VovIsLm
VovIsNc	
File	vovutils.tcl
Usage	VovIsNc
VovIsWin	
File	vovutils.tcl
Usage	VovIsWin
VovJobClassGetProperty	
File	vovutils.tcl
Usage	VovJobClassGetProperty className propName defaultValue (default=)

VovJSON::getArray

File vovjson.tcl

Usage VovJSON::getArray args

VovJSON::getObject

File vovjson.tcl

Usage VovJSON::getObject args

VovJSON::init

File vovjson.tcl

Usage VovJSON::init

VovLDAP::close

File vovldap.tcl

Usage VovLDAP::close lh

VovLDAP::getDnAttributes

File vovldap.tcl

Usage VovLDAP::getDnAttributes dn attrList (default=)

Description Get attributes for a specific DN.

VovLDAP::getEmail

File vovldap.tcl

Usage VovLDAP::getEmail users delim (default=) subject
(default=) body (default=)

Description Get e-mail addresses from LDAP based on user names. Requires one or more users to be passed in, separated by commas.

Return User name if none is found. If a subject is passed in, return a mailto link for users with subject. Otherwise, return a list of e-mail addresses.

VovLDAP::getGroupMembers

File vovldap.tcl

Usage VovLDAP::getGroupMembers group

Description Get members of a specific group

VovLDAP::getUserAttributes

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::getUserAttributes user attrList (default=)</code>
Description	Get a list of attributes for a specific user. If no attributes are passed, use config file settings.

VovLDAP::getUserDn

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::getUserDn user</code>
Description	Get the DN for a specific user.

VovLDAP::getUserGroups

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::getUserGroups user</code>
Description	Get groups to which a specific user belongs

VovLDAP::getUsersMatching

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::getUsersMatching attr value</code>
Description	Get users matching a specific attribute value

VovLDAP::init

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::init</code>

VovLDAP::map

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::map results</code>
Description	Order and map results according to configuration

VovLDAP::open

File	<code>vovldap.tcl</code>
Usage	<code>VovLDAP::open</code>

VovLicenseViolationCallback

File	<code>vov_boot.tcl</code>
Usage	<code>VovLicenseViolationCallback hasViolation</code>

VovLimitClientsForThisUser

File	<code>vov_common.tcl</code>
Usage	<code>VovLimitClientsForThisUser nickname max</code>
Description	Find how many clients with the give nickname we already have.

vovLongId

File	<code>vov_common.tcl</code>
Usage	<code>vovLongId id</code>

VovLooksLikeAnId

File	<code>vovutils.tcl</code>
Usage	<code>VovLooksLikeAnId s</code>

VovMessage

File	<code>vovutils.tcl</code>
Usage	<code>VovMessage msg verboseLevel (default=0)</code>

VovMetricsReloadHistorical

File	<code>vovtraceutils.tcl</code>
Usage	<code>VovMetricsReloadHistorical family beginTs</code>
Description	This procedure reloads historical metrics for the given family back to the specified timestamp

VovOutputConflictHandlerAbort

File	<code>vovfdl.tcl</code>
Usage	<code>VovOutputConflictHandlerAbort jobId placeId oldJobId</code>

VovOutputConflictHandlerRetry

File	<code>vovfdl.tcl</code>
Usage	<code>VovOutputConflictHandlerRetry jobId placeId oldJobId</code>

VovParseStdOptions

File	<code>vovutils.tcl</code>
Usage	<code>VovParseStdOptions argList usage allowEmptyArgList (default=0)</code>
Description	Parse options -h and -v Also print usage message if the argList is empty.

VovParseTimeSpec

File	<code>vovutils.tcl</code>
Usage	<code>VovParseTimeSpec spec</code>
Description	Convert a VOV timespec into a number of seconds The empty string returns 0. Bad specs cause an error.

VovPlotMetrics

File	<code>vovmetricplots.tcl</code>
Usage	<code>VovPlotMetrics args</code>

VovPlotMetricsFrame

File	<code>vovmetricplots.tcl</code>
Usage	<code>VovPlotMetricsFrame frame script args</code>

VovPlotMetricsFrame_close

File	<code>vovmetricplots.tcl</code>
Usage	<code>VovPlotMetricsFrame_close frame</code>

VovPlotMetricsFrameSetConfig

File	<code>vovmetricplots.tcl</code>
Usage	<code>VovPlotMetricsFrameSetConfig frame element val</code>

VovPlotShowSchedulerMetrics

File	<code>vovmetricplots.tcl</code>
Usage	<code>VovPlotShowSchedulerMetrics topFrame configFile</code>

VovPredictGet

File	<code>vovfdl.tcl</code>
-------------	-------------------------

Usage VovPredictGet

VovPredictRam

File vovfdl.tcl

Usage VovPredictRam

VovPredictReduce

File vovfdl.tcl

Usage VovPredictReduce str max (default=80)

Description Convert to string

VovPredictXdur

File vovfdl.tcl

Usage VovPredictXdur

VovPrintStack

File vovfdl.tcl

Usage VovPrintStack

VovPrintUsage

File vovutils.tcl

Usage VovPrintUsage message

VovProduct

File vovutils.tcl

Usage VovProduct args

VovProjectStall

File vovtraceutils.tcl

Usage VovProjectStall whyReason

VovProjectUnstall

File vovtraceutils.tcl

Usage VovProjectUnstall whyReason

VovPsig

File	<code>vovutils.tcl</code>
Usage	<code>VovPsig args</code>
Description	Compute the signature of an Altair Engineering project from the environment (default) or from positional or keyword parameters
	Examples:
	<ul style="list-style-type: none">• VovPsig: compute signature from env-vars• VovPsig proj host: proj@host• VovPsig -host foo -proj bar -port 666: bar@foo:666

vovReadFile

File	<code>vovenvutils.tcl</code>
Usage	<code>vovReadFile file</code>

VovReorderTags

File	<code>vov_policy.tcl</code>
Usage	<code>VovReorderTags inputTagList referenceTagList</code>
Description	Given an input list of tags (inputTagList) and a partial order of tags specified in referenceTagList, return a list of all input tags in an order that satisfies the partial ordering.

VovRgyFindEntries

File	<code>vovregistrylib.tcl</code>
Usage	<code>VovRgyFindEntries avar args</code>
Description	Scan array named by avar (loaded by VovRgyLoadData) for entries matching any of the -attr options

VovRgyGetDirectory

File	<code>vovregistrylib.tcl</code>
Usage	<code>VovRgyGetDirectory</code>

VovRgyGetGeneric

File	<code>vovregistrylib.tcl</code>
Usage	<code>VovRgyGetGeneric avar psig key</code>

VovRgyLoadData

File	vovregistrylib.tcl
Usage	VovRgyLoadData namepat userpat avar verbose (default=0)
Description	Fill variable named by avar with registry info; return count of entries If namepat is non-null-string, include only entries matching name regex If user is non-null-string, include only entries owned by user.

VovScanClock

File	vovutils.tcl
Usage	VovScanClock s
Description	Best effort to scan a clock specification. It could also be a unix timestamp
Examples:	

```
VovScanClock 20081231
VovScanClock 20081231T120000
VovScanClock [clock seconds]
VovScanClock "Oct 31, 2008"
VovScanClock "31-Oct-2008"
```

VovSearchFile

File	vovutils.tcl
Usage	VovSearchFile file dirList errorHandling (default=quiet)

VovServerBusyCallback

File	vov_boot.tcl
Usage	VovServerBusyCallback

VovServerConfig

File	vovtraceutils.tcl
Usage	VovServerConfig name value
Description	<p>This procedure is used mostly with the output from "vovshow - policy"</p> <p>Another version of this procedure is defined by vovserver before sourcing the policy.tcl file</p>

VovServerMessage

File	vovutils.tcl
Usage	VovServerMessage msg

VovSetCGIError

File	vov_boot.tcl
Usage	VovSetCGIError msg

VovSetChangeStatusProp

File	vovtraceutils.tcl
Usage	VovSetChangeStatusProp id msg status
Description	Keep a history of reconciled

vovSetGlobalBackground

File	vov_boot.tcl
Usage	vovSetGlobalBackground projectName bg (default=) fg (default=)
Description	<p>This procedure is called from vov_boot.tcl There is a duplicate in tracesrv.cc (VovTrace::getTraceColor()). This should call the C++ procedure. It is possible to have many VOV windows associated to different projects. We want to distinguish them using the background and foreground color. This procedure sets the global window background and foreground to be a function of the project name. If the variables VOV_BACKGROUND and/or VOV_FOREGROUND are set, they prevail. The default colors in VOV are designed to work with a light background.</p>
Return	The pair (foregroundColor, backgroundColor).
Side Effects	vov_color(background) and vov_color(foreground) are set.

vovShortId

File	vov_common.tcl
Usage	vovShortId id
Description	Convert id from its long to its short form. For example, 00012345 becomes 12345.

VovShowProgress

File	vovfdl.tcl
Usage	VovShowProgress module (default=10)

VovSignature

File	vovutils.tcl
Usage	VovSignature

vovTaskerLoadColorMap

File	vov_common.tcl
Usage	vovTaskerLoadColorMap

vovTaskerStatus2Color

File	vov_common.tcl
Usage	vovTaskerStatus2Color status

vovTaskerStatus2ColorFg

File	vov_common.tcl
Usage	vovTaskerStatus2ColorFg status

VovSQL::canonicalizeLogName

File	vovsql.tcl
Usage	VovSQL::canonicalizeLogName logFile

VovSQL::canUseMaterializedView

File	vovsql.tcl
Usage	VovSQL::canUseMaterializedView handle view start_ts end_ts

VovSQL::checkMySqlConnection

File	vovsql.tcl
Usage	VovSQL::checkMySqlConnection

VovSQL::close

File	vovsql.tcl
-------------	------------

Usage `VovSQL::close handle`

Description Closes a connection handle

VovSQL::commit

File `vovsql.tcl`

Usage `VovSQL::commit handle`

Description OBSOLETE

Commits statements that have been processed when auto commit mode is off

VovSQL::configure

File `vovsql.tcl`

Usage `VovSQL::configure args`

Description Gets database configuration from db/config.tcl.

VovSQL::createIndex

File `vovsql.tcl`

Usage `VovSQL::createIndex handle table index columns args`

Description Create an index

VovSQL::createMaterializedView

File `vovsql.tcl`

Usage `VovSQL::createMaterializedView handle table view`

VovSQL::createSequence

File `vovsql.tcl`

Usage `VovSQL::createSequence handle name args`

Description OBSOLETE

Create an Oracle sequence, do nothing for other vendors

VovSQL::createTable

File `vovsql.tcl`

Usage `VovSQL::createTable handle table members`

VovSQL::createTempTable

File vovsql.tcl

Usage VovSQL::createTempTable handle table members

VovSQL::createTrigger

File vovsql.tcl

Usage VovSQL::createTrigger handle table name when what sql args

Description Create a trigger that executes the specified SQL

VovSQL::dropIndex

File vovsql.tcl

Usage VovSQL::dropIndex handle table index args

Description Drop an index

VovSQL::escapeString

File vovsql.tcl

Usage VovSQL::escapeString handle str maxLen (default=0)

VovSQL::exec

File vovsql.tcl

Usage VovSQL::exec handle stmt args

Description Execute a non-select statement Return the number of affected rows or 0.

VovSQL::getColumnCollation

File vovsql.tcl

Usage VovSQL::getColumnCollation handle table column args

Description Return the collation of the column.

VovSQL::getColumnDescription

File vovsql.tcl

Usage VovSQL::getColumnDescription handle table column

Description Return the description of the column or the empty string if the column does not exist.

VovSQL::getColumnNumericPrecision

File vovsql.tcl

Usage VovSQL::getColumnNumericPrecision handle table column
args

Description Return the numeric precision of the specified column for the specified table.

VovSQL::getColumns

File vovsql.tcl

Usage VovSQL::getColumns handle table args

Description Return a list of columns found in the specified table.

VovSQL::getColumnWidth

File vovsql.tcl

Usage VovSQL::getColumnWidth handle table column args

Description Return the width of the specified column for the specified table.

VovSQL::getDbApp

File vovsql.tcl

Usage VovSQL::getDbApp

Description Return the value of variable VOV_DB_APP.

VovSQL::getDbDir

File vovsql.tcl

Usage VovSQL::getDbDir

Description Get database directory

VovSQL::getDbInfo

File vovsql.tcl

Usage VovSQL::getDbInfo

VovSQL::getFeatureIdNameMap

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getFeatureIdNameMap handle array args</code>
Description	Used for loading the feature ID to name mapping and vice-versa filtered by any specified tag IDs -tagids { idList } -featureids { idList } -featurenames { nameList } -negatetags Constrain the lookup to the NOT of the specified tags -negatefeatures Constrain the lookup to the NOT of the specified features

VovSQL::getIdNameMap

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getIdNameMap handle table array args</code>
Description	Loads reference tables for ID to name mapping and vice-versa Args: -ids { idList } -names { nameList } -negate Constrain the lookup to the NOT of the specified object names

VovSQL::getLogFileId

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getLogFileId handle relFile</code>
Description	Returns the ID of a file that has been loaded into the database

VovSQL::getPassword

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getPassword u</code>

VovSQL::getTagForFeatureId

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getTagForFeatureId handle feature_id</code>
Description	Returns tag associated with a feature Id

VovSQL::getTrackedFileData

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getTrackedFileData handle fileName forceReload dataVar</code>

VovSQL::getVendor

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::getVendor</code>
Description	Returns the configured or overridden DB vendor

VovSQL::init

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::init</code>

VovSQL::isColumn

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::isColumn handle table column args</code>
Description	Determine if a column exists in the specified table

VovSQL::isConstraint

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::isConstraint handle table constraint args</code>
Description	Determine if a constraint exists in the specified table

VovSQL::isDatabase

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::isDatabase handle dbname</code>

VovSQL::isIndex

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::isIndex handle table index args</code>
Description	OBSOLETE except for <code>vovsql_create_tables_jobs</code> , which should create indexes along the way instead of at the end. Determine if an index exists in the configured database

VovSQL::isSequence

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::isSequence handle sequence args</code>

Description OBSOLETE Determine if an Oracle sequence or trigger exists. Returns 1 for other vendors since they do not use sequences.

VovSQL::isTable

File vovsql.tcl

Usage VovSQL::isTable handle table args

Description Determine if a table exists in the configured database

VovSQL::isTrigger

File vovsql.tcl

Usage VovSQL::isTrigger handle trigger args

Description Determine if a trigger exists

VovSQL::isUser

File vovsql.tcl

Usage VovSQL::isUser handle user args

Description Determine if a user exists in the configured database

VovSQL::lastInserted

File vovsql.tcl

Usage VovSQL::lastInserted handle table

Description Returns the id of the last inserted record in the table

VovSQL::loop

File vovsql.tcl

Usage VovSQL::loop handle stmt variables script args

Description Execute Tcl upon each row in the result set

VovSQL::open

File vovsql.tcl

Usage VovSQL::open args

Description Opens and returns a connection handle for the configured database

VovSQL::query

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::query handle stmt args</code>
Description	Execute a select statement

VovSQL::raiseError

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::raiseError msg info (default=) code (default=-1)</code>

VovSQL::refreshMaterializedView

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::refreshMaterializedView handle view</code>

VovSQL::rollback

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::rollback handle</code>
Description	<p>OBSOLETE</p> <p>Rolls back changes made by statements that have been processed when auto commit mode is off</p>

VovSQL::setAutoIncrement

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::setAutoIncrement handle table value</code>
Description	Sets the auto increment value for a table to the specified value.

VovSQL::setErrorControl

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::setErrorControl ec</code>
Description	Set the way errors are handled in the VovSQL namespace. See <code>raiseError</code> for possible values. Note that "internal" is used only during the sourcing of the sql config file, to differentiate between an error raised on purpose, such as "Database is not running," and an error in the actual sourcing of the file.

VovSQL::setPassword

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::setPassword u p</code>

VovSQL::showConfig

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::showConfig</code>
Description	Show configuration

VovSQL::sqlite3_busy_callback

File	<code>vovsql.tcl</code>
Usage	<code>VovSQL::sqlite3_busy_callback</code>
Description	Procedure called if the sqlite3 database is locked.

VovSQLPort::autoIncrement

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::autoIncrement</code>
Description	Returns SQL fragment for defining a column that auto-increments its values. This is applicable to MySQL and SQLite only. Oracle uses a sequence and trigger to accomplish this.

VovSQLPort::autoInteger

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::autoInteger args</code>
Description	Returns SQL fragment for the data type for an auto-incrementing integer This is applicable to PostgreSQL only, other vendors are forwarded to the <code>VovSQLPort::integer</code> procedure as normal.

VovSQLPort::begin

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::begin</code>
Description	OBSOLETE Returns SQL fragment for BEGIN

VovSQLPort::char

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::char width</code>
Description	Returns a SQL fragment for defining a character column type of the specified width

VovSQLPort::concat

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::concat args</code>

VovSQLPort::createTempTable

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::createTempTable</code>
Description	Returns a SQL fragment for creating a temporary table

VovSQLPort::dropTempTable

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::dropTempTable table</code>
Description	Returns a SQL statement for dropping a temporary table

VovSQLPort::formatTs

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::formatTs field format</code>
Description	Heatmap uses day,hour format

VovSQLPort::greaterThan

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::greaterThan a b</code>
Description	Returns a SQL fragment returning a boolean bit based on whether value a is greater than value b.

VovSQLPort::greatest

File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::greatest</code>

Description Returns a SQL fragment for the vendor-specific function that compares column or static values to determine the one with the highest value.

VovSQLPort::init

File vovsql.tcl

Usage VovSQLPort::init

VovSQLPort::integer

File vovsql.tcl

Usage VovSQLPort::integer type (default=INT) signed (default=UNSIGNED)

Description Return SQL fragment containing integer data type definition based on arguments passed.

VovSQLPort::isNull

File vovsql.tcl

Usage VovSQLPort::isNull

Description Returns SQL fragment for use in a where clause to determine if a column contains a null value.

VovSQLPort::least

File vovsql.tcl

Usage VovSQLPort::least

Description Returns SQL fragment for the vendor-specific function that compares column or static values to determine the one with the lowest value.

VovSQLPort::lessThan

File vovsql.tcl

Usage VovSQLPort::lessThan a b

Description Returns SQL fragment returning a boolean bit based on whether value a is less than value b.

VovSQLPort::roundTs

File vovsql.tcl

Usage	<code>VovSQLPort::roundTs field format roundTo</code>
Description	Denial plot uses this, but only Oracle requires it. The formatTs procedure above will work for all other vendors. This will be OBSOLETE once we pull support for Oracle
VovSQLPort::vendor	
File	<code>vovsql.tcl</code>
Usage	<code>VovSQLPort::vendor vendorList sql alt (default=)</code>
Description	Returns the specified SQL if the configured vendor matches that of the specified vendor. Otherwise, it returns the alternate SQL if specified.
VovStartRetraceHierarchical	
File	<code>vovsql.tcl</code>
Usage	<code>VovStartRetraceHierarchical priority mode resources what targetId recursiveLevel (default=0)</code>
VovTmpFile	
File	<code>vovsql.tcl</code>
Usage	<code>VovTmpFile file</code>
VovTranslateMsgText	
File	<code>vovutils.tcl</code>
Usage	<code>VovTranslateMsgText msg</code>
Description	Temporarily just return message for English. When i18n or l10n is added, this is where the translation logic will be added.
VovUpdateValidityWave	
File	<code>vovutils.tcl</code>
Usage	<code>VovUpdateValidityWave daemon now verbose (default=0)</code>
Description	A validity wave is a wave that has values u and 1
VovUserError	
File	<code>vovutils.tcl</code>
Usage	<code>VovUserError msg noexit (default=0)</code>

vovUseSnapshot

File	vovutils.tcl
Usage	vovUseSnapshot
Description	We use the snapshot if: the env(VOV_ENV) is not set. the env(VOV_ENV) is set to either "" or "DEFAULT" or if it begins with the component "SNAPSHOT" with no file argument (i.e. no SNAPSHOT(...)).

VovVerboseMessage

File	vovutils.tcl
Usage	VovVerboseMessage verbose msg

VovVersion

File	vovutils.tcl
Usage	VovVersion
Description	Return the version of the current VOVDIR installation. It expects to find the file \$VOVDIR/etc/version.txt

VovWarning

File	vovutils.tcl
Usage	VovWarning msg verboseLevelIgnored (default=0)

VovWhy

File	vovwhy.tcl
Usage	VovWhy nodeId args

VovWXWhy

File	vovwhy.tcl
Usage	VovWXWhy jobId status

vtk_counter_create

File	vovtraceutils.tcl
Usage	vtk_counter_create name

vtk_fairshare_get

File	<code>vov_policy.tcl</code>
Usage	<code>vtk_fairshare_get</code>
Description	Added for backwards compatibility.

vtk_flexlm_exclude_tags

File	<code>vov_policy.tcl</code>
Usage	<code>vtk_flexlm_exclude_tags tags</code>

vtk_flexlm_monitor_filter

File	<code>vov_policy.tcl</code>
Usage	<code>vtk_flexlm_monitor_filter tag feature</code>
Description	Filter features for <code>vtk_flexlm_monitor_all</code> based on tag and feature name. The procedure must return a Boolean. The default procedure simply returns 1 for all features.

vtk_flexlm_monitor_mapname

File	<code>vov_policy.tcl</code>
Usage	<code>vtk_flexlm_monitor_mapname tag feature</code>
Description	<p>This is a procedure that can be overwritten in <code>resources.tcl</code>.</p> <p>Map license feature name to FT resourcemap right-hand-side The default procedure simply returns the empty string, meaning no mapping.</p>

vtk_flexlm_monitor_resname

File	<code>vov_policy.tcl</code>
Usage	<code>vtk_flexlm_monitor_resname tag feature</code>
Description	Map license tag and feature name to VOV resourcemap name. The default procedure simply prepends License: to the feature name

vtk_flexlm_turbo

File	<code>vov_policy.tcl</code>
Usage	<code>vtk_flexlm_turbo lmFeature turboFactor</code>
Description	Use <code>vtk_flexlm_overbook</code> instead.

VTK_GENERIC_GET_BUCKETS

File vovtraceutils.tcl

Usage VTK_GENERIC_GET_BUCKETS array

vtk_percent_pp

File vtkutils.tcl

Usage vtk_percent_pp number

vtk_resource_policy_hook

File vov_policy.tcl

Usage vtk_resource_policy_hook projectName resName max

Description This procedure is used to limit the max number of a resource assigned to a project.

vtk_resourcemap_procedure

File vov_policy.tcl

Usage vtk_resourcemap_procedure procedure

Description Register a procedure used to update a resource map. Only used in vovresourced.

vtk_resourcemap_sum

File vov_policy.tcl

Usage vtk_resourcemap_sum sum components

Description Define a resource as being the sum of some components. The resource is then also mapped to the OR of the components. The counterpart of this procedure is updateAllSumResources which is called by vovresourced.

vtk_trace_info

File vovtraceutils.tcl

Usage vtk_trace_info args

vtk_user_info

File vovtraceutils.tcl

Usage vtk_user_info args

W

WITHOUT_TRACING

File	<code>vovfdl.tcl</code>
Usage	<code>WITHOUT_TRACING script</code>
Description	This command is useful to execute a script while disabling all established runtime tracing.

wxGetConnectionInfo

File	<code>vovwxconnect.tcl</code>
Usage	<code>wxGetConnectionInfo connectionInfo</code>

wxSetupForLSF

File	<code>vovwxconnect.tcl</code>
Usage	<code>wxGetConnectionInfo connectionInfo wxSetupForLSF NCQEMUL</code>

wxSetupForNC

File	<code>vovwxconnect.tcl</code>
Usage	<code>wxSetupForNC queues</code>

wxSetupGE

File	<code>vovwxconnect.tcl</code>
Usage	<code>wxSetupGE</code>

wxSetupLSF_Leg

File	<code>vovwxconnect.tcl</code>
Usage	<code>wxSetupLSF_Leg emulationOnly ncQueue</code>
Description	Setup vovlsfd

wxSetupNC_Leg

File	<code>vovwxconnect.tcl</code>
Usage	<code>wxSetupNC_Leg qn args</code>
Description	Setup vovelasticd

wxSetupPBS

File `vovwxconnect.tcl`

Usage `wxSetupPBS`

wxSetupPBS_Leg

File `vovwxconnect.tcl`

Usage `wxSetupPBS_Leg emulationOnly ncQueue`

wxShowConnections

File `vovwxconnect.tcl`

Usage `wxShowConnections`

wxTestConnections

File `vovwxconnect.tcl`

Usage `wxTestConnections`

wxUpdateConfigParams

File `vovwxconnect.tcl`

Usage `wxUpdateConfigParams configFile oldParams newParams`

X

X

File `vovfdl.tcl`

Usage `X xdur`

Description Set default expected duration for the jobs to follow. The argument is a valid time specification, such as "10" or "2m" or "1h30m"

Side Effects `make(xdur)` is set.

X11_DISPLAY

File `vovfdl.tcl`

Usage `X11_DISPLAY value objectSpec (default=)`

Description Set the X11_DISPLAY property on an object. This property is used by the jobs that have a runmode of `xterm_icon` or `xterm_open`.

The value is a valid X11 DISPLAY that can be used to open X windows. If the value is empty, the property is deleted. The objectSpec can be -help to show a usage message, empty or "main" to set the property globally, or "lastjob" to set it on the most recent job.

Examples:

```
X11_DISPLAY localhost:0.0
X11_DISPLAY localhost:0.0
X11_DISPLAY localhost:0.0 main
X11_DISPLAY mac01:0.0 lastjob
X11_DISPLAY -help
```

Z

File	vovfdl.tcl
Usage	Z xdur
Description	Make all files in list zippable (i.e. set the compressable flag). Utility to set the zippable flag if necessary.

VTK API Available in vovtasker

The binary vovtasker has access to only a limited VTK API. This is because the protocol between vovtasker and vovserver has special requirements. This limited API is used, for example, in the taskerLSF.tcl file, in taskerRes.tcl, and in general in all Tcl scripts read by vovtasker.

vtk_tasker_change_field FIELD VALUE	The field is one of taskergroup capacity model release
vtk_tasker_suspend MESSAGE	Suspend the tasker, set the message.
vtk_tasker_resume TIMELEFT MSG	TIMELEFT is a timespec.
vtk_tasker_set_timeleft TIMESPEC	TIMESPEC is a timespec or the work "unlimited" (case-insensitive)
vtk_tasker_set_prop OBJID PROPNAME PROPVALUE	Set a property on an object. The property is always of string type. There is no feedback on whether the action was successful.

vtk_tasker_job_control ACTION JOBID	ACTION is one of STOP, SUSPEND, RESUME, while JOBID can be the jobId of a running job or 0 to mean "all jobs".
vtk_tasker_job_started JOBID STARTTIME	Used by indirect taskers. Report to vovserver that a job has started. This is useful but not required. It allows the vovserver to know more quickly about the job start.
vtk_tasker_job_recover JOBID	
vtk_tasker_job_stats JOBID STAT1 VALUE1 ...	
vtk_idle_time_get	This uses vovxidle to capture the information on the idle time of display :0.0.
vtk_fs_stat PATH INFO	Get information about the file system that contains PATH. The info that is returned is specified by INFO which could be one of bsize, blocks, bfree bavail, files, or fsid.

VTK Procedures

vtk_acl

vtk_acl_op

Usage

Usage	vtk_acl_op objectId command agent actionList
Description	This procedure is used to manage object ACLs.
AGENTS:	
ADMIN	- All users that have been assigned as project administrators
LEADER	- All users that have been assigned as project leaders
EVERYBODY	- All users in the system
OWNER	- The project owner
HOST <name>	- All users coming from a specific host
USER <name>	- A specific user
OSGROUP <name>	- All users in a specific OS group
FSGROUP <name>	- All users in a specific FairShare group
USERGROUP <name>	- All users in a specific user group
COMMANDS:	
GET	- Get current ACLs on an object
RESET	- Reset ACLs on an object to defaults
APPEND	- Appends ACLs to existing ACLs an object
SET	- Sets ACLs on an object. Existing ACLs are replaced.
DELETE	- Delete an ACL action or element from an object.
TEST	- Test if the object grants permissions to the given user.
EXAMPLES:	
vtk_acl_op 012345 GET	- Get current ACLs on 012345
vtk_acl_op 012345 RESET	- Reset ACLs on 012345 to defaults
vtk_acl_op 012345 APPEND ADMIN { RETRACE STOP }	- All admin users can retrace or stop 012345
vtk_acl_op 012345 APPEND HOST pluto { SUSPEND }	- All users on host pluto can suspend 012345
vtk_acl_op 012345 APPEND USER john DELEGATE	- User john can assign ACLs on 012345
vtk_acl_op 012345 APPEND USER john ALL	- User john can do all.
vtk_acl_op 012345 APPEND OSGROUP proj_a VIEW	- All users in OS group proj_a can view properties on 012345
ACTIONS:	
ATTACH	- Create a relationship between objects
DETACH	- Destroy a relationship between objects
EDIT	- Edit properties of an object
VIEW	- View properties of an object
RETRACE	- Retrace an object

```
STOP      - Stop an object
SUSPEND   - Suspend an object
FORGET     - Forget an object
DELEGATE   - Assign ACLs on an object
```

Returns string showing modified ACL,
or an error string, one of "Illegal id", "Acl Denied" "Unsecured object"

vtk_ae_getkeysfromfile

Usage

```
vtk_ae_getkeysfromfile [filename]
```

Description

A vtk wrapper for Vov_AE_GetKeysFromFile, to read a key pair from an existing file and return them.

Opens a file that contains a secret/public keypair, reads them, and returns them.

If a filename is passed, that file is attempted to be read.

If no filename is passed, then

\$HOME/.vov/userkey or %USERPROFILE%\vov\userkey is read.

Seealso

Vov_AE_GetKeysFromFile

Returns

A list with the first entry being the secret key and the second entry being the public key

Examples

```
set keypair [vtk_ae_getkeysfromfile]
```

```
set seckey [lindex $keypair 0]
```

```
set pubkey [lindex $keypair 1]
```

vtk_alert

vtk_alert_ack

Usage

```
vtk_alert_ack id
```

Description

Acknowledge an alert by its ID

vtk_alert_add

Usage

```
vtk_alert_add level title [OPTIONS]
```

Description

Where level is one of URGENT ERROR WARN or INFO

Options:

- *detail info -- Add more detail to the alert.
- *module modulename -- Specify the name of the module reporting the alert.
- *thresh threshold -- Number of times alert must be added before it is visible.
- *idvar idVar -- Name of variable to set with alert ID. If procedure succeeds but idVar is 0, the vovserver is an older version that does not return alert IDs. Add an alert to the system. Only the first letter of 'level' is used (U,E,W,I) The word CRITICAL is the same as ERROR and it is still supported for backwards compatibility.

Returns

Visible or invisible

vtk_alert_forget

Usage

```
vtk_alert_forget id
```

Description

Delete an alert from the system by its ID

vtk_alert_set_defaults

Usage

```
vtk_alert_set_defaults -module MODULENAME
```

Description

Set the module name for all subsequent alerts.

vtk_annotation

vtk_annotation_add

Usage

```
vtk_annotation_add objectId note_text
```

Description

Add the annotation 'note_text' to the object specified by 'objectId'.

Example

```
vtk_annotation_add 12345 "This is an annotation"
```

Returns

Nothing on success. An error message on failure.

vtk_annotation_count

Usage

```
vtk_annotation_count objectId
```

Description

Get the number of annotations attached to the object with id equal to 'objectId'.

Returns

The number of annotations.

vtk_annotation_delete

Usage

```
vtk_annotation_delete annotationId
```

Description

Delete the annotation with ID equal to 'annotationId'.
To retrieve the ID of an annotation you can use 'vtk_annotation_get'.

Returns

Nothing on success. An error message on failure.

vtk_annotation_get

Usage

```
vtk_annotation_get objectId noteIndex
```

Description

Returns: the noteIndexth annotation associated to the given object.
If either the object or the note index don't exist, then an error is returned.
A list of four (4) elements is returned.

- * annotationId
- * timestamp
- * signature
- * content

Returns

nodeId, timestamp, signature, content

vtk_annotation_modify

Usage

```
vtk_annotation_modify annotationId noteText
```

Description

Set the contents of the annotation with Id equal to 'annotationID' to the string given by 'noteText'.
To retrieve the ID of an annotation, see the method vtk_annotation_get.

Returns

Nothing

vtk_artificial

vtk_artificial_dependency_declare

Usage

```
vtk_artificial_dependency_declare jobId jobOrSetId2 [flags]
```

Description

```
vtk_artificial_dependency_declare 1234 1334
```

vtk_asynch

vtk_asynch_notify_set_filter

Obsolete

Usage

```
vtk_asynch_notify_set_filter CLEAR or: (APPEND | REPLACE) subject verb -[RX rx | ID id]
```

Description

Set a filter for the current asynchronous communication channel.
This allows a substantial reduction of the number of event that get sent to the channel.

Note

This procedure is obsolete and its use is discouraged.
Use vtk_eventfilter_append and vtk_eventfilter_clear instead.

See Also

[vtk_eventfilter_append](#)

vtk_bar

vtk_bar_chart

Usage

```
vtk_bar_chart [OPTIONS] <DATALIST>
```

Description

Generate a GIF file representing either a bar chart. The data can be in the form NUMBER or NUMBER:LABEL. If no label is shown, then the empty label is used.

Options:

```
* -w      N           Width of image (400 pixels)
* -h      N           Height of image (400 pixels)
* -d              Print data instead of label on X axis
* -s      N           Limit Display data: to N data items, sum remainder in data[N+1]
* -t      X           Create Tcl array named as N containing color map for entries
* -f      X           Name of output GIF file ("out.gif")
* -title   X           Specify a title to be drawn at the top of the image
* -xtitle  x           Specify a title to be drawn for the x axis
* -ytitle  xtitle      Specify a title to be drawn for the y axis
* -firstred           Force first bar to be red
* -allred            Force all bars to be red
* -allgreen          Force all bars to be green (-firstred takes precedence)
* -allblue           Force all bars to be blue (-firstred takes precedence)
```

Examples:

```
vtk_bar_chart -w 800 -h 400 -f mygraph.gif 10 20 30 44 55
vtk_bar_chart -w 800 -h 400 -f mygraph.gif 10:us 20:eu 30:ch 44:jp 55:ru
```

vtk_base64

vtk_base64_decode

Usage

```
vtk_base64_decode string
```

Description

Decode a base64-encoded string.

vtk_base64_encode

Usage
vtk_base64_encode string

Description
Encode string using base64 encoding.

vtk_cache

vtk_cache_clear

Usage
vtk_cache_clear

Description
Clear the local cache of nodes and sets held by the GUI.

Example
vtk_cache_clear

See Also

[vtk_gui](#)

vtk_chain

vtk_chain_get

Usage
vtk_chain_get placeId

Description
If place is part of a chain, return the list of ids of all places in the chain, in the correct order.
If place is not a chain, just return its id.
If the id is not of a place, return error.

Returns
A list of ids.

vtk_checkout

vtk_checkout_checkin

Usage:

```
vtk_checkout_checkin VovId
```

Description:

Explicitly check-in a handle in the VOV build-in license emulator. This procedure is rarely used, because most jobs simply exit.

Example:

```
vtk_checkout_checkin 002345
```

vtk_checkout_create

Usage:

```
vtk_checkout_create featureId user host tty handle tokens coTs ciTs [OPTIONS]
```

Description:

Create or update a feature user. If the handle is the same as another current feature user, then that record is updated, else a new record is created. Used to keep track of license checkouts for external license daemons like flexlm.

Options:

- * -pid pid
- * -queued 0|1
- * -color color
- * -checkout2 checkout2 -- The checkout timestamp reported by lmstat
- * -account account
- * -linger lingerTime
- * -version versionOfCheckedoutFeature

Returns:

The id of the feature user.

vtk_checkout_delete

Usage:

```
vtk_checkout_delete VovId
```

Description:

Delete a handle from the license emulator. Requires ADMIN privileges. The argument represents the VovId of a license handle.

Example:
vtk_checkout_delete 001234556

vtk_checkout_get

Usage:
vtk_checkout_get id array [-renew]

Description:
Get information about a checkout user (also called a "handle" or a "checkout"). The -renew option is only needed when calling this procedure under the name vtkle_checkout_get, and its role is to tell the license emulator to check if any queued request can be promoted to a checkout.

vtk_checkout_set

Usage:
vtk_checkout_set id array

Description:
Set information about a feature user (also called a "handle" or a "checkout").

vtk_checkouts

vtk_checkouts_get

Usage
vtk_checkouts_get array [OPTIONS]

Description
List all checkouts managed by the current vovserver.

Options:
* -tag list
* -feature list
* -host list
* -user list
* -account list
* -max N
* -limit N
* -case BOOL

```
* -nocase -- Same as -case 0
```

Examples

```
setenv VOV_JOBCOUNTER 5555@bear,licmon  
vtk_checkouts_get chkInfo -tag "MGC CDN" -user "john mary"  
parray chkInfo
```

vtk_client

vtk_client_log_genkey

Usage

```
vtk_client_log_genkey none
```

Description

Generates client log key.

vtk_client_log_getkey

Usage

```
vtk_client_log_getkey none
```

Description

Reads client log key from env var.

vtk_client_log_login

Usage

```
vtk_client_log_login none
```

Description

Logs into message queue server.

vtk_client_mgr

Usage

```
vtk_client_mgr
```

Description

Secret command to manage attachments on the client side.

vtk_client_name

Usage

```
vtk_client_name name
```

Description

Assign a name to this client.

A name is typically a short string that helps identify the client.

Long names are truncated to 30 characters.

The name cannot begin with a dash.

Example

```
vtk_client_name vovnotifyd
```

Returns

Nothing

See Also

[vtk_clients_get](#)

vtk_client_version

Usage

```
vtk_client_version version
```

Description

Assign a version to this client, typically something like 2015.03, truncated to 30 chars.

Example

```
vtk_client_version 2015.03
```

Returns

Nothing

See Also

[vtk_clients_get](#)

vtk_clients

vtk_clients_get

Usage

```
vtk_clients_get arrayname [OPTIONS]
```

Description

Retrieve information about the current clients to vovserver.

1. -name NAME, only get clients with given name

2. -user NAME, only get clients with given user
3. -host NAME, only get clients with given host

Examples

```
vtk_clients_get clientInfo  
vtk_clients_get clientInfo -name notifyd  
vtk_clients_get clientInfo -user john
```

Returns

Fills given array with information about clients.

vtk_cmd

vtk_cmd_quote_for_shell

Usage

```
vtk_cmd_quote_for_shell
```

Description

Add quotes to a command so it can be executed by a shell

vtk_crc

vtk_crc string

Usage

```
vtk_crc string
```

Description

Calculate CRC of specified string.

Optional, specify a modulus to reduce the resulting value (2-100000).

vtk_dailylog

vtk_dailylog_close

Usage

```
vtk_daily_close id
```


Description

Close a daily log opened by a previous call to `vtk_dailylog_open`.

Examples

```
set dl [vtk_dailylog_open "logs/%s.log"]  
vtk_dailylog_write $dl "hello world"  
vtk_dailylog_close $dl
```

vtk_dailylog_open

Usage

```
vtk_dailylog_open pattern [-compress] [-noheader] [-sympath path]
```

Description

Open a new daily log at `pattern`, replacing `%s` in `pattern` with `YYYY.MM.DD`, and return its identifier. Optionally compresses previous day's log at rotation, and/or skips the boilerplate header. `sympath` is a symbolic link (a shortcut on windows) name created to the latest log after rotation.

Examples

```
set dl [vtk_dailylog_open "logs/%s.log"]  
vtk_dailylog_write $dl "hello world"  
vtk_dailylog_close $dl
```

vtk_dailylog_write

Usage

```
vtk_dailylog_write id message ?newline?
```

Description

Write message to the daily log represented by `id`. Include a `"\n"` if `newline` is non-zero (which is the default).

Examples

```
set dl [vtk_dailylog_open "logs/%s.log"]  
vtk_dailylog_write $dl "hello world"  
vtk_dailylog_close $dl
```

vtk_date

vtk_date_format

Obsolete

```
Usage
vtk_date_format timestamp

Description
OBSOLETE: use clock format instead.
The argument 'timestamp' is the usual absolute time value.
It represents then number of seconds elapsed since 00:00:00 on January 1, 1970,
Coordinated Universal Time (UTC).

Returns
A representation of the timestamp in the form "Sat Jun 25 13:01:23 2011"
```

vtk_date_pp

Obsolete

```
Usage
vtk_date_pp date

Description
OBSOLETE: use clock format instead.
Pretty-print the date.

Example
vtk_date_pp [clock] seconds]
Fri Apr 4 12:04:08 2020
```

vtk_debug

```
Usage
vtk_debug [flagname] [flagvalue]

Description
This procedure controls some debugging flags on the client side.
With no parameters, it shows all the flags.
With two parameters, the string flagname and the boolean flagvalue, it sets the flag
'flagname' to the value 'flagvalue'.

Examples
vtk_debug
```

```
Notification      0
VovBuffer         0
Bsocket          0
EventFilter       0
LogicalNames      0
Resources         0
vtk_debug Notification 1
```

vtk_debug_set

Usage

```
vtk_debug_set flags
```

Description

Set the "trace" flags to debug server/client communication.
Look into vil.h for definition of these flags
The setting only affects the current client.

Examples

```
vtk_debug_set 16      -- activate RPC codes
vtk_debug_set 0       -- clear the flags
```

vtk_disable

Usage

```
vtk_disable
```

Description

This procedure is used in safe Tcl interpreters to disabled
selected procedures and commands that are considered dangerous from
a security point of view.

Typically, the following Tcl commands are disabled
in a safe Tcl interpreter:
cd close exec exit fcopy open send.

Returns

A warning message.

Examples

```
vtk_disable
```

vtk_equivalence

vtk_equivalence

Usage

```
vtk_equivalence logicalName physicalPath
```

Description

This procedure is used only in the equiv.tcl file of a project. The procedure has several functions:

1. Define an equivalence between a logical name and a physical name.

Example

```
vtk_equivalence TOP /projects/ram  
vtk_equivalence TOP /remote/export/projects/ram
```

2. Control the capitalization of path names. In this case, the procedure needs one argument which is either `-nocase` or `-case`. With `-nocase`, all path names are canonicalized to lower case.

This is useful if the VOV server is running on a Windows machine.

Example

```
vtk_equivalence -nocase; # If server runs on Windows
```

3. This can also be used to control whether AFS paths should be normalized or not. Use option `-afs` to normalize (e.g. `/.automount/hostname/xxx` becomes `/net/hostname/xxx`) and option `-noafs` to leave those paths alone (the default).

Example

```
vtk_equivalence -afs; # Normalize AFS paths.
```

Side Effects

1. The environment variable corresponding to the logical name is set, if it does not exist already.
2. The Tcl global variable corresponding to the logical name is set to the value of the environment variable.

vtk_equivalence_get_cache

Usage `vtk_equivalence_get_cache hostname`

Description

This procedure is used to get the cache of equivalence rules for a given host.

See documentation on equivalences for more information.

If the entry for `_default_` is defined, it will be returned for any host that does not have a specific entry.

If `hostname` is the null string `""`, the procedure returns the list of hosts for which equivalences have been defined.

Example

```
set listOfHosts [vtk_equivalence_get_cache ""]
```

```
set rulesForHost [vtk_equivalence_get_cache "pluto"]
```

vtk_equivalence_set_cache

Usage

```
vtk_equivalence_set_cache hostname values
```

Description

This procedure is used to set the cache of equivalence rules for a given host. See documentation on equivalences for more information. If the values field is empty, the entry will be deleted. If the hostname field is "_default_", then the default value will be defined.

Examples

```
vtk_equivalence_set_cache "pluto" "LICDIR /export/licenses"  
vtk_equivalence_set_cache "_default_" "LICDIR /export/licenses"
```

vtk_equivalences

vtk_equivalences_debug

Usage

```
vtk_equivalences_debug 1|0
```

Description

Control debugging level in equivalences code.

Examples

```
vtk_equivalences_debug 1  
vtk_equivalences_debug 0
```

Returns

Nothing

vtk_equivalences_find

Usage

```
vtk_equivalences_find path, array.
```

Example

```
vtk_equivalences_find . equiv_array
```

```
parray equiv_array  
/export/units/abc  
E:/units/abc
```

Returns

Find all the equivalences for the given path and put them into the array.

vtk_equivalences_get

Usage

```
vtk_equivalences_get resultArray
```

Example

```
vtk_equivalences_get equivInfo
```

Returns

Fills result array with information about equivalences.

vtk_equivalences_refresh

Usage

```
vtk_equivalences_refresh
```

Description

This procedure forces the rereading of the equiv.tcl file.

Example

```
vtk_equivalences_refresh
```

Returns

Nothing

vtk_equivalences_show

Usage

```
vtk_equivalences_show
```

Description

This procedure prints the table of equivalences.

Example

```
vtk_equivalences_show  
Directory of equiv files: .  
Executable equiv file:    /Users/john/vov/mac81.swd/equiv.tcl  
Cached      equiv file:    /Users/john/vov/mac81.swd/equiv.caches/mac05
```

```
VOVDIR      -> /Users/john/rtda/2009.04mac0512/macosex
VOVDIR      -> /Users/john/rtda/2009.04mac0512/common
* BUILD_TOP -> /Users/john/projects/mac81
* HOME      -> /Users/john
```

Returns
Nothing

vtk_event

vtk_event_control

Usage

```
vtk_event_control flag
```

Description

This procedure controls the asynchronous channel of communication with the VOV server (the "Channel").

This procedure normally is used to START and STOP the Channel, but it can also be used to temporarily SUSPEND the Channel and then to RESUME it.

The procedure also controls whether events are passed to the TCL interpreter (on the client side) by means of the two commands START_TCL_HANDLER and STOP_TCL_HANDLER. 'flag' is one of:

- * INFO get info about local event buffer (current event count, hi-water mark)
- * STATUS get status of connection
- * START to open the channel
- * STOP to close the channel
- * SUSPEND to temporarily close the channel
- * IGNORE to temporarily ignore events
- * HANDLE to handle events again (opposite of IGNORE)
- * RESUME to reopen a suspended channel
- * START_TCL_HANDLER to start dispatching events to the Tcl handler (expensive)
- * STOP_TCL_HANDLER to stop dispatching events to the Tcl handler.
- * DISCONNECT_HANDLER_NONE if disconnected, do nothing
- * DISCONNECT_HANDLER_TCL if disconnected, call tcl handler
vovDroppedAsyncConnectionCallBack
- * DISCONNECT_HANDLER_RECONNECT if disconnected, simply reconnect
- * UPDATE_ALL to update all graph objects in the console

Example

```
vtk_event_control START
vtk_event_control START 2000;      # Start with a max buf size of 2000MB
vtk_event_control SUSPEND
vtk_event_control RESUME
vtk_event_control STATUS
vtk_event_control INFO
vtk_event_control DISCONNECT_HANDLER_TCL
```

Returns
Nothing.

vtk_event_generate

Usage

```
vtk_event_generate numberOfEvents "...subjectList..." "...verbList..."
```

Description

Generate the specified number of events with all the combinations of subjects and verbs.

This procedure is used only in Flowtracer testing.

Examples

```
vtk_event_generate 100 JOB INFORM generate 100 events (job, inform).
```

```
vtk_event_generate 10 NODE {CHANGE MOVE} generate 10 events (node, change) and 10 events (node, move).
```

```
vtk_event_generate 1 ALL INFORM generate 1 event (subject, inform) for each possible subject: (annotation, inform), (block, inform), (tasker, inform) etc.
```

```
vtk_event_generate 2 {NODE BLOCK FLOW} {CHANGE MOVE} generate 2 events for each possible couple.
```

```
vtk_event_generate 1 ALL ALL generate every possible event.
```

Note

If you generate random events (like in the last example) it could happen that most of them does not have any sense (for example: (JOURNAL, MOVE)).

vtk_event_get

Usage:

```
vtk_event_get [timeout (default 60)] [resultArray]
```

Description:

When called with up to 1 argument, this procedure returns a list corresponding to an event.

When called with 2 arguments, the procedure fills the resultArray with information about an event.

If no event is available, the procedure waits for up to 'timeout' seconds and then returns with value "TIMEOUT".

Example:

```
set ev [vtk_event_get 5]
```

Set the variable 'ev' with the next event.

If no events are available in the next 5 seconds then it returns "TIMEOUT"

Example:

```
set ev [vtk_event_get 30 resArray]
```

if an event is available it is loaded into the array 'resArray'. In case of timeout the variable ev is set to the value "TIMEOUT"

Returns:

A list corresponding to an event in either of the following two formats.

subject verb id aux

subject verb id timestamp message

Or the string `WOULDBLOCK` or the string `TIMEOUT`.
This list is loaded on the array 'resultArray' if it is present.

vtk_event_hurl

Usage
`vtk_event_hurl eventId subject verb "description"`

Description
Generate the specified event.
This procedure is used mostly in VOV testing.

Examples
`vtk_event_hurl 4721 JOB STOP ""`
`vtk_event_hurl 6354 NODE MOVE "23 6"`
`vtk_event_hurl 6354 NODE GUIUPDATE`

Note:
With this `vtk` you could generate events that do not make any sense (for example: (JOURNAL, MOVE)).

vtk_event_subjects_map

Usage:
`vtk_event_subjects_map arrayname`

Description:
Fills the array with subject's character and the meaning for all events.

Example:
`vtk_event_subjects_map myarray`
`parray mymap`
`mymap(a) = ANNOTATATION`
`mymap(d) = ALVE`
`...`

vtk_event_verbs_map

Usage:
`vtk_event_verbs_map arrayname`

Description:
Fills the array with verb's character and the meaning for all events.

```
Example:  
vtk_event_verbs_map myarray  
parray mymap  
mymap(a) = ATTACH  
mymap(b) = BARRIER  
...
```

vtk_eventfilter

vtk_eventfilter_append

```
Usage:  
vtk_eventfilter_append subject_list verb_list [-RX regExp | -ID id]
```

Description:
Set a filter for the current asynchronous communication channel.
This allows a substantial reduction of the number of event that get sent to the channel.

Examples:
vtk_eventfilter_append JOB {START DISPATCH ERROR STUCK}
...now only the events (job, start), (job, dispatch), (job, error) and (job, stuck) are sent to the channel.

vtk_eventfilter_append FILE INFORM
...now also the events (file, inform) are sent.

vtk_eventfilter_append RETRACE STOP
...and (retrace, stop) too.

Note:
This command work as an "OR" filter i.e. an event is sent to the channel if it pass at least one of the filters in the chain.
To clear the chain use the command vtk_eventfilter_clear.

vtk_eventfilter_clear

```
Usage:  
vtk_eventfilter_clear
```

Description:
Reset the filter for the current asynchronous communication channel. It does not require any arguments. Use it as it is: vtk_eventfilter_clear

vtk_eventmonitor

Usage

```
vtk_eventmonitor textWidget
```

Description

This procedure implements the event monitor window in the VOV console. It is designed to be used by the GUI developer, not by the regular user.

Examples

```
vtk_eventmonitor textWidget
vtk_eventmonitor textWidget config
vtk_eventmonitor textWidget config -mode
vtk_eventmonitor textWidget config -mode normal|active|disabled
vtk_eventmonitor textWidget config -autoscroll
vtk_eventmonitor textWidget config -autoscroll <BOOLEAN>
vtk_eventmonitor textWidget config -maxlines
vtk_eventmonitor textWidget config -maxlines <INTEGER>
vtk_eventmonitor textWidget config -filter
vtk_eventmonitor textWidget config -filter subject_name1 verb_name1 subject_name2
  verb_name2 ...
vtk_eventmonitor textWidget insert <MESSAGE>
vtk_eventmonitor textWidget reset
vtk_eventmonitor textWidget destroy
```

vtk_eventwidget

Usage:

```
vtk_eventwidget Events widget
```

Description:

This procedure enable/disable the printing of the events received by the GUI. It is designed to be used by the GUI developer, not by the regular user.

Examples:

```
vtk_eventwidget <widget>
vtk_eventwidget <widget> enable // Events printing: enabled.
vtk_eventwidget <widget> disable // Events printing: disabled.
vtk_eventwidget <widget> destroy // Destroy the widget.
```

vtk_exclude

vtk_exclude_rule

Usage:

```
vtk_exclude_rule [-prefix|-regexp|-clear|-tool toolrx|-jobclass jobclassrx|-jobname  
jobnamerx] string
```

Description:

Define a rule to exclude files from the trace. This procedure is used mostly the exclude.tcl file but it can also be used inside of capsules. The rule can be a prefix or a regular expression. If you use the -clear option, which cancels all rules, the string argument is ignored. This procedure is normally used in the exclude.tcl file in the server configuration directory. It can also be used in the encapsulation script of a tool.

Examples:

```
vtk_exclude_rule -clear "ignored_arg"  
vtk_exclude_rule -prefix /var/nis  
vtk_exclude_rule -regexp {.*Depend.state$}
```

Returns:

Nothing

See Also

[vtk_path_is_excluded](#)

vtk_exit

Usage:

```
vtk_exit exitStatus
```

Description:

This procedure calls VovEnd(exitStatus), therefore closing cleanly the communication with the VOV server. The exitStatus is an integer in the range [0-255].

The procedure, which is useful only in instrumented Tcl scripts in place of exit, does not return.

Example:

```
vtk_exit 0
```

vtk_feature

vtk_feature_checkout

Usage:

```
vtk_feature_checkout featureId [OPTIONS]
```

Description:

Procedure to check-out a certain number of tokens of a feature using the VOV built-in license server emulator. This procedure is also called `vtkle_feature_checkout`.

Options:

* -queue	If not enough tokens are available, queue the request.
* -now	If not enough tokens are available, fail.
* -tokens <N>	Specify number of tokens (positive, default 1).
* -account Account	Specify account (like LM_PROJECT or RLM_PROJECT).

vtk_feature_delete

Usage:

```
vtk_feature_delete id
```

Description:

Destroy a feature from the VOV built-in license emulator.

Example:

```
vtk_feature_delete 0012344
```

vtk_feature_find

Usage:

```
vtk_feature_find daemonId name [array]
```

Description:

Find a feature for a daemon by name. Returns the VovId of the feature or 0 if the feature cannot be found.

If the third argument 'array' is provided, the array is filled with the information about the feature.

vtk_feature_get

Usage:

```
vtk_feature_get id array
```

Description:
Get information about a feature. Fill given array with the information.

vtk_feature_get_or_create

Usage:
vtk_feature_get_or_create daemonId name total version

Description:
Create or modify a feature with a specified name, total capacity and version.

Note:
This procedure should be renamed to vtk_feature_create_or_modify

Example:
vtk_feature_get_or_create \$daemonId "my_spice" 22 "1.0" "matlab"
000123456

Returns:
The id of feature

vtk_feature_get_version_pool

No information for this procedure.

vtk_feature_set

Usage:
vtk_feature_set featureId array

Description:
Update information about a feature in the VOV built-in license emulator.
Only expire, version, and total can be updated.

(This procedure should be renamed to vtk_feature_modify).

Example:
set info(expire) [expr \$now + 100000]
set info(total) 123
vtk_feature_set \$featureId info

Returns:
Nothing

vtk_feature_set_version_pool

No information for this procedure.

vtk_features

vtk_features_get

```
Usage:
vtk_features_get array [OPTIONS]

Description:
List all features managed by the current flowtracer server. The server must
be specified in the VOV_JOBCOUNTER variable which has the form "PORT:HOST" or
"PORT:HOST,NAME". OPTIONS:

* -tag list
* -isv list
* -feature list
* -checkouts -- Show also checkouts.

Example:
setenv VOV_JOBCOUNTER 5555@bear,licmon
vtk_features_get featureInfo -tag MGC
parray featureInfo
```

vtk_file

vtk_file_get

```
Usage:
vtk_file_get <remote> <local>

Description:
File staging procedure (experimental). Get a file from the server over to the local
client.
The remote file must physically be in the "staging area" of the vovserver, which is
typically
NAME.swd/staging or can be controlled in vovserver with the environment variable
VOV_STAGING_DIR.

Example:
vtk_file_get 282981019201982019289028.tmp options.lic
```

vtk_file_put

Usage:

```
vtk_file_put <local> <remote>
```

Description:

File staging procedure (experimental). Put a file from the local client onto server. The remote file must physically be in the "staging area" of the vovserver, which is typically NAME.swd/staging or can be controlled in vovserver with the environment variable VOV_STAGING_DIR.

Example:

```
vtk_file_put options.lic 282981019201982019289028.tmp
```

vtk_file_read

No information for this procedure.

vtk_filesystem

vtk_filesystem_get

Usage:

```
vtk_filesystem_get fullpath array.
```

Description:

This procedure retrieves information (into the array) about the filesystem that contains the file.

Example:

```
vtk_filesystem_get /remote/release/vov/current/scripts/INSTALL.sh fsinfo parray  
fsinfo  
...array output...
```

vtk_flexlm

vtk_flexlm_monitor

Usage

```
vtk_flexlm_monitor [OPTIONS] [TAG/]lmFeature [vovResource [vovMap]]
```

Description

Declare a FLEXlm feature to be monitored.

The first argument is the name of the FLEXlm feature
The feature may be tagged or untagged. An example of tagged feature is "SNPS/
VCSRuntime_Net",
while the untagged version would be "VCSRuntime_Net"

The second argument (optional) is the name of the VOV resource
corresponding to the FLEXlm feature. By default,
the vovResource is the same as the lmFeature with the prefix License:.

The third argument (optional) is the map for the resource. By default
the map is empty.

By default, if TAG is not specified, all tags that have
the named feature are aggregated. This can be changed
by the following options.

Options:

```
-excludetags  list-of-tags-to-exclude
-includetags  same as -tags below
-tags         list-of-tags-to-include
-order        order-of-tags
-noooq                -- Disable Out-Of-Queue computation
```

vtk_flexlm_monitor_all

Usage

```
vtk_flexlm_monitor_all [options]
```

Description

Monitor multiple FLEXlm features, using data from Monitor.
The default action with no options is to create resourcemap
for all features monitored by Monitor.
Use this procedure with caution, because it may suddenly add
thousands of resource maps to your project.

Options:

```
-daemon <host:port>  Specify host:port for Monitor.
-fproc  <procname>    Name of procedure to filter tagged features.
-rproc  <procname>    Name of procedure to map feature names to resource names.
-mproc  <procname>    Name of proc to map feature names to resourcemap right-hand-
side.
-tag     <tagname>     Append to list of tags to search; default is all tags.
-tags    <list>        As above
-If      <regex>       Append regex to list of feature include patterns.
-Xf      <regex>       Append regex to list of feature exclude patterns.
-It      <regex>       Append regex to list of tag include patterns.
-Xt      <regex>       Append regex to list of tag exclude patterns.
-order   <list of tags> Specify partial order between tags. May be repeated.
-sum1    Even if there is only one tag for a feature, use the sum
operation.
```

Returns

Number of features monitored

vtk_flexlm_overbook

Usage

```
vtk_flexlm_overbook lmFeature OPTIONS
```

Description

Options

-factor f	-- Control boost. Default 1.0
-thresh t	-- The fraction of the total above which overbooking is activated. Default 0.9
-headroom int	-- How many FLEXlm licenses to leave alone.
-queued int	-- How many FLEXlm licenses can be queued (above this, overbooking stops).
-enable bool	-- Set to 0 to just print what overbooking would do
-verbose bool	-- Set to 1 to print more info about the overbooking operation

vtk_foreign_bucket_reset

Usage

```
vtk_foreign_bucket_reset bucketId
```

Description

This procedure is used to "hard reset" the number of jobs in a foreign bucket (imported from a WX queue) object from server.

Setting the number of jobs to 0 will cause the bucket to be auto-deleted.

Returns

"ok" or an error message

Example

```
vtk_foreign_bucket_reset 00001234 1000  
ok
```

vtk_fs

vtk_fs_latency

No information for this procedure.

vtk_fs_stat

Usage:

```
vtk_fs_stat path [which]
```

Description:

The old name for this procedure was `vtk_statvfs`.
This procedure retrieves information about a file system.
The filesystem is identified by the `path`.
The value of `which` determines which information is returned

Value of 'which'	Data retrieved
""	free space in MB
bsize	block size
blocks	number of blocks
bfree	number of free blocks
bavail	number of available blocks
files	number of files
fsid	File System Id

vtk_fsgroup

vtk_fsgroup_create

Usage:

```
vtk_fsgroup_create fullname [-weight w] [-window ts]
```

Description:

Creates the fairshare group named 'fullname' if it does not exist.
The options `-weight` and `-window` specify the weight and time window of the fsgroup.
If absent, default values are assigned. Example: If `/time/regression` does not exist before this VTK call, it will be created along with `/time/regression/projectA` `vtk_fsgroup_create /time/regression/projectA`

Example:

```
vtk_fsgroup_create /time/regression/projectA
```

Returns:

VOV id of created fairshare group or the string 'nochange' if the fsgroup already exists.

Note:

If `fullname` contains components that do not exist, they are created automatically, with default values for their weight and time window. When the fsgroup exists, `nochange` is returned even when a different weight or window is given, and the fsgroup's values are unchanged.

vtk_fsgroup_delete

Usage:
vtk_fsgroup_delete fsgid

Description:
Deletes the fairshare group having VOV id 'fsgid' and BEWARE! all its subordinates.

Returns:
VOV status code

vtk_fsgroup_find

Usage:
vtk_fsgroup_find fullname arrayname

Description:
Retrieves information into the array named by 'arrayname' about the fairshare group named 'fullname'

Returns:
VOV id of fairshare group if found, or 0 if there is no such fairshare group

Note:
In either case, the array element 'found' is set to 1 or 0 respectively if the fairshare group exists or does not.

Example:
vtk_fsgroup_find /time/users/jchen info
parray info
...array output...

vtk_fsgroup_get

Usage:
vtk_fsgroup_get id arrayname

Description:
Retrieves information into the array named by 'arrayname' about the FairShare group named 'fullname'

Returns:
VOV id of FairShare group if found, or 0 if there is no such FairShare group

vtk_fsgroup_set

Usage:

```
vtk_fsgroup_set fsgid arrayname
```

Description:

Sets the values of the FairShare group having VOV id 'fsgid' to those in the array named by 'arrayname'. Only the array slots 'weight', 'window', and 'owner' are {used;} the values of other array slots are ignored.

Returns:

VOV status code

vtk_fsgroup_update

Usage:

```
vtk_fsgroup_update
```

Description:

Forces the server to update the statistics of the FairShare groups, such as Target Shares, Actual Shares, Excess Shares. Calling this function leaves a message in server log that says Updated fairshare statistics.

Returns:

VOV status code

vtk_ftlm

vtk_ftlm_agent_all

Usage:

```
vtk_ftlm_agent_all
```

Description:

Define an ftlm_agent tasker for all license daemons known to licmon. Similar to vtk_flexlm_monitor_all

vtk_ftlm_agent_define

Usage:
vtk_ftlm_agent_define

Description:
Define an agent tasker for licmon

vtk_generate_ae_keypair

Usage
vtk_generate_ae_keypair

Description
Generate a public/secret keypair used for API Keys.
Generates a public/secret keypair and returns it as a list.
The first entry is the secret key and the 2nd is the public key.

Returns
A list with the first entry being the secret key and the second entry being the public key

vtk_generic

vtk_generic_get

Usage:
vtk_generic_get key array

Description:
This procedure fills the given array with information based on the value of key:

Value of key	Retrieved Information
project	project name, status, port, total nodes,
taskers, ...	
fields	all fields
buckets	the buckets in the job queue
fairshare	the fairshare status
dates	the clock offsets of the various hosts
resourcemaps	the resource maps
features	the features (licenses) managed by this
server	
hosts	the hosts
filesystems	the filesystems
resourcelist	the list of all resources

retracelist	the list of all retracing
users	the list of all users
alerts	the list of all alerts
preemptrules	the list of all preemption rules
preemptpools	the list of all preemption pools
queries	the list of all queries

Examples

```
vtk_generic_get resourcemaps resinfo  
vtk_generic_get fairshare fairshareinfo
```

Returns:

'ok' or 'error'

vtk_geo

vtk_geo_get

No information for this procedure.

vtk_geo_set

Usage:

```
vtk_geo_set
```

Description:

The second argument looks like { {22 10 20 3 4 } { 23 20 20 3 4 } ... } i.e. a list of quintuplets.

vtk_get

vtk_get_nctasker_res

Usage

Get the union of the resources offered by the NC taskers.
For consumable res, return the sum of those on each tasker.

vtk_graph

Usage:
vtk_graph canvasPath cmd ...

Description:

This procedure is the interface to the widget that implements the GUI graph. It is not meant to be used by the casual user. The first argument is the Tk path to a canvas widget, while cmd is one of:

```
config -progresslabel <labelpath>
config -mode <grid> <list> <jobs> <graph>
config -update <flag>                -- Enable/disable asynch update.
config -showFlags [<flag>]            -- Enable/disable colorizing of flags.
config -id [<setId>]                  -- Set the id of the set to draw.
config -incrementalRegion
x0 y0 x1 y1                          -- Set the region to draw incremental nodes
config -redrawLimit {nodeCount}       -- If node count is below this, do a full redraw
  when new nodes are added
destroy
clean
doplacement -graph                   -- Do graph placement
doplacement -geo                     -- Do geo placement
doplacement -any                     -- Do geo placement if it is {available;}
  otherwise graph
doplacement -grid                   -- Do grid placement
geo save                             -- Save current placement
geo get ID setgeo ID X Y W H
draw <setId> <fitFlag>
drawlabel
redraw                               -- Redraw without recomputing the draw
  parameters.
initialize <width> <height>          -- Set some parameter for drawing.
expand <nodeId> <neighborsTooFlag>
autofit                             -- Request a delayed autofit
hide <filename> [off]               -- Hide/Restore(==off) the nodes with similar
  name.
info widget                          -- Get info about the widget.
info id                              -- Current set ID.
info iteminfo                        -- For the given item, get the associated short
  information, which is the command line
                                     for jobs and the name for files.
info status                          -- Return valid or invalid
info stopped                         -- Return if drawing was stopped ( 1, 0 )
info graphmode                       -- Return the current graph mode (list, grid,
  graph, jobs)
info selection                       -- Get the list of the id of the nodes currently
  selected.
info selectionbyitem                 -- Get the list of the itemid of the nodes
  currently selected.
info drawnarea                       -- Obsolete.
info id2item                         -- Get the main item for the given id info
id2items                            -- Get all items for the given id
info item2id info item2label info item2status -- Unused.
info stats
info statistics
info isempty                         -- Return 1 if no set has never been drawn, 0
  otherwise.
```



```
info item2ptr          -- debug purpose: it's for the gui developer, not
  for the user.
info gcsiz             -- Get the number of objects drawn in the graphic
  context notify -- Obsolete.
navigate <forward|backward>
lowernode <nodeId>
raisenode <nodeId>
removenode <nodeId>
stop                  -- Interrupt drawing of current graph.
nostop               -- reset stop drawing of current graph.
zoom <scaleFactor>
scale x y zoomFactorX zoomFactorY -- Update widget scaling factors
select all           -- Select all the items on the canvas
select clear         -- Unselect all the items on the canvas
select invert        -- Invert the selection on the canvas
select <itemId>
select <itemId> <flag>
highlight <itemId>   -- Get info about the highlighting status of the
  node.
highlight <itemId> <flag> -- Enable/disable highlighting of the node.
listviews
setview <viewname>    -- Set graph view by name.
```

vtk_groupname

Usage:
vtk_groupname

Description:
Get the name of the current UNIX operating system group.

Example:
vtk_groupname integration

Returns:
A string

See Also

[vtk_logname](#)

vtk_gui

vtk_gui_refresh

Usage:
vtk_gui_refresh

Description:
Clear the local cache, redraw all the open canvas and re-cache their contents (set).

Example:

```
vtk_gui_refresh
```

Returns:
Nothing

See Also

[vtk_cache_clear](#)

vtk_gzip

vtk_gzip_read

Usage:
`vtk_gzip_read file.gz length`

Description:
Read a gzipped file, up to the given length

Example:
`vtk_gzip_read cpu.v.gz 1000`

Returns:
The content of the file, up to "length" bytes.

vtk_hash

vtk_hash_name

Usage:
`vtk_hash_name string`

Description:
Hash the input string into an integer in the range [0-65535].
This procedure is used, for example, to compute the default background color of a project based on the project name.

Example:
`vtk_hash_name abc`
49158

Returns:
An integer.

vtk_host

vtk_host_forget

Usage:
vtk_host_forget id

Description:
Delete a host from the system by its ID

vtk_host_id

Usage:
vtk_host_id [-first | -all]

Description:
This procedure returns the hostId of the current host as it is used by FlowTracer for licensing purposes. This is either the host id or the MAC address, depending on the architecture. On hosts with multiple MAC addresses, you can use the option -all to see them all.

Examples:
On Linux
vtk_host_id
MAC:00:02:b3:64:29:28
vtk_host_id -all
MAC:00:02:b3:64:29:28 MAC:00:02:b3:64:31:33

Returns:
A string representing the host id

vtk_hostname

Usage:
vtk_hostname none

Description:
Get the name of the current host. This works on both UNIX and Windows.

Example:
vtk_hostname
bigmac

Returns:
A string

See Also

[vtk_groupname](#)

[vtk_user_get_shell](#)

vtk_init

Usage:

```
vtk_init level
```

Description:

This procedure is used, for example, in vovconsole. This procedure is used by scripts that want to use some of the GUI dialogs. The procedure initialize the TCL, TK, and TIX packages. The argument level has to be one of "tcl", "tk", "tix" or "all". Each level implies all the preceding levels.

The procedure can be called multiple times. The argument "tcl" does nothing because TCL is always initialized. The typical value for level is "all" and is used by scripts that want to pop up a dialog.

Examples:

```
vtk_init all
```

Returns:

Nothing

vtk_input

vtk_input_declare

Usage:

```
vtk_input_declare transId placeId [flags]
```

Description:

Declare an input dependency between a transition and a place. This is used in a Flow Description (See FDL procedure I). The procedure returns an error if the given ids are not valid or the new dependency creates a cycle.

Flags can be either -normal, -consumed or -sticky.

Example:

```
vtk_input_declare 12345 4567  
vtk_input_declare 12345 4567 -normal  
vtk_input_declare 12345 67890 -consumed  
vtk_input_declare 12345 67890 -sticky
```

vtk_input_declare_ex

Usage:

```
vtk_input_declare_ex transId placeName [options]
```

Description:

An alternative to `vtk_input_declare`, which uses 2 VovIds, while this uses the name of the output.

Declare an input dependency between a transition and a place. This is used in a Flow Description

(See FDL procedure I).

Options

- * -normal
- * -consumed
- * -nocheck
- * -sticky
- * -links(which is ignored in this context)
- * -db DB

Examples:

```
vtk_input_declare_ex 12345 abc
vtk_input_declare_ex 12345 abc -normal
vtk_input_declare_ex 12345 abc -consumed
vtk_input_declare_ex 12345 abc -nocheck
vtk_input_declare_ex 12345 abc -sticky
```

Returns:

The procedure returns an error if the given `transId` is not valid the new dependency creates a cycle

See Also

[vtk_nodes](#)

vtk_integer

vtk_integer_pp

Usage:

```
vtk_integer_pp integer
```

Description:

Pretty-print an integer by separating the thousands with commas.

Examples:

```
vtk_integer_pp 1000
1,000
vtk_integer_pp 1234567
1,234,567
```

vtk_job

vtk_job_control

Usage

```
vtk_job_control taskerId action jobId opt1 opt2 opt3 -why TEXT_REASON taskerId
STOP jobId [Options] -why TEXT_REASON taskerId SUSPEND jobId [Options] -why
TEXT_REASON
```

Description

This is the main procedure to send controlling signals and modifications to jobs running on remote taskers. The argument taskerId can either be a legal VovId, "ALL", or the number 0 which is equivalent to "ALL".

The argument action can be one of STOP KILL DEQUEUE SUSPEND RESUME SIGUSR1 SIGUSR2 SIGTSTP CHECK EXT MODIFY

- If jobId is 0, then all jobs on the specified taskers are affected.
- If jobId is the string "TASKER", then the tasker is stopped gracefully.
- If jobId is the string "TASKER/FORCE", then the taskers is stopped with force. Taskers can only be stopped.

The optional argument -why "Optional Reason" can be used to pass a reason to the WHYSTATUS field and property of affected nodes.

Options for action EXT, MODIFY:

Table 4:

Options	Action EXT	Action MODIFY
opt1	signalName	fieldName
opt2	procNameIncludeRx	newValue
opt3	procNameExcludeRx	unused

Options for action STOP,SUSPEND :

- -signals comma-separated list of signals
- -include comma-separated list of process names to include in sending signals
- -exclude comma-separated list of process names to exclude in sending signals
- -delay delays in seconds between sending the list of signals



Note:

- STOP and KILL are equivalent
- The include and exclude lists used by STOP and SUSPEND are mutually exclusive; if both are specified the exclude list will apply.
- DEQUEUE does not reach the tasker and is processed by vovserver
- CHECK forces the taskers to scan the process table and gather information about the processes and send it to the vovserver
- EXT uses the script vovjobctrl to execute the job control. Check the documentation about vovjobctrl for more information about this type of job control.

Examples

```
# Stop job 23344:
vtk_job_control 0 STOP 23344 -why "Custom script stopped job"
# Stop all jobs on taskers 21221:
vtk_job_control 21221 STOP 0
# Modify autokill on running job 24455 to 120s:
vtk_job_control 0 MODIFY 24455 autokill 120 -why "Autokill job from job control"
# Suspend all sleep jobs using SIGINT followed by SIGHUP on tasker 12345:
vtk_job_control 12345 SUSPEND -signals INT,HUP -include sleep
# Kill all jobs except sleep jobs using the default signal cascade with 2
seconds between signals on tasker 23456:
vtk_job_control 23456 STOP -exclude sleep -delay 2
```

vtk_jobclass



Note: Some jobclass features, such as idle timeout, require the notification service to be enabled for the feature to work properly.

vtk_jobclass_set Autoforget

Usage:

```
vtk_jobclass_set Autoforget jobclass autoforgetSpec
```

Description:

Set the autoforget time. If autoforget is not positive, the autoforget property is deleted.

Examples:

```
vtk_jobclass_set Autoforget "hsim_lo" "2m"
vtk_jobclass_set Autoforget "hsim_lo" "0"
```

vtk_jobclass_set_idle_delays

Usage:

```
vtk_jobclass_set_idle_delays jobclass warnDelay killDelay
```

Description:

Set the delays for warning and or killing jobs in the class that are using no CPU time.

vtk_jobclass_set_max_reschedule

Usage:

```
vtk_jobclass_set_max_reschedule jobclass maxRescheduleNumber
```

Description:

Set maximum number of rescheduling that can be done on a job belonging to the specified class.

Example:

```
vtk_jobclass_set_max_reschedule hsim 1
```

vtk_jobclass_set_preemption_method

Usage:

```
vtk_jobclass_set_preemption_method jobclass preemptionMethod
```

Description:

Set the default preemption method for the jobs in a jobclass.

Example:

```
vtk_jobclass_set_preemption_method "hsim_lo" "STOP"
```

vtk_jobclass_set_revocation_delay

Usage:

```
vtk_jobclass_set_revocation_delay jobclass timespec
```

Description:

Set the revocation delay for a jobclass. When a job in the class has been running for specified delay, all resources of the job that are in the condition "Declared/NotUsed" are revoked from the job and made available for the general pool.

vtk_jobqueue

vtk_jobqueue

```
Usage:
vtk_jobqueue cmd args

Description:
'cmd' is one of:

- info <jobId> <option>      -- Get info about a job
- refresh <n>                 -- Refresh the cached database
- count                       -- Count the jobs in the queue
- get                         -- Get the list of the jobs in the queue
- time, timepp               -- Total time of jobs in the queue.
- unknown                     -- Number of jobs of unknown duration in the queue.

Note:
Do not use this procedure. Use vtk_jobqueue_get instead.
```

vtk_jobqueue_delete

```
Usage:
vtk_jobqueue_delete jobId|setId ... -why WHY_REASON

Description:
Delete one or more jobs from the queue. This has the effect of descheduling the given jobs. If the ids refer to sets, then all jobs in the set are descheduled. The WHY_REASON is optional. If included, gets set on WHYSTATUS field of job.

Examples:
vtk_jobqueue_delete 12122 12124
vtk_jobqueue_delete 12122 12124 -why {Custom script stopped jobs}

Returns:
Nothing
```

vtk_jobqueue_get

```
Usage:
vtk_jobqueue_get [n] | ["buckets"] | "count"

Description:
Get information about the job queue. If option n is

1. equal to 0, the procedure returns 3 numbers.
   1. number of jobs in the queue
   2. total expected duration
```

```
3. number of unknown jobs

2. greater than 0, it returns a verbose report, with 'n' indicating the number of
   jobs to report (OBSOLETE: DO NOT USE)

3. less than 0, return list of buckets. Use vtk_generic_get_buckets instead.

If the argument is the string 'count' the procedure returns just the total number of
jobs in the queue.

If the argument is the string 'buckets', the procedure returns a list with one
element for each bucket. Each element in turn is a list with the following elements:

1. Number of jobs in the bucket.
2. Group
3. User
4. Tool
5. Priority
6. Expected duration
7. A quoted list of resources
```

See Also

[vtk_generic_get](#)

[vtk_select_loop](#)

vtk_la_ctrl

```
Usage
vtk_la_ctrl Action

Description
Control LA operations by requesting it take the specified action.
Currently, the only available action is: "config_reread". This must be performed
BEFORE the config.tcl file is read, and it MUST be followed by reading the
config.tcl file.

Requires ADMIN privileges and reserved for use by License Allocator daemon that
implements user commands.

Examples:
vtk_la_ctrl config_read
```

vtk_licdaemon

vtk_licdaemon_delete

Usage:
vtk_licdaemon_delete id

Description:
Destroy a license daemon.

vtk_licdaemon_find

Usage:
vtk_licdaemon_find tag host port

Description:
Find a license daemon by triplet (tag,host,port).

Return:
The id of the daemon or 0 if not found.

vtk_licdaemon_get

Usage:
vtk_licdaemon_get id array

Description:
Get information about a license daemon.

vtk_licdaemon_get_or_create

Usage:
vtk_licdaemon_get_or_create

Description:
Get or create a license daemon data structure.

Return: the id of the daemon

vtk_licdaemon_set

Usage:
vtk_licdaemon_set

Description:
To modify a license daemon.

vtk_licdaemon_update

Usage:
vtk_licdaemon_update

Description:
Label all daemons with tag 'tag' as being updated.

vtk_licdaemons

vtk_licdaemons_find_by_tag

No information for this procedure.

vtk_licdaemons_get

Usage:
vtk_licdaemons_get

Description:
Get the list of all license daemons known to this flowtracer server.

vtk_license

vtk_license_check

Usage:

```
vtk_license_check mode
```

Description:

This procedure asks the server to check the license. The argument mode is case insensitive and is one of "CHECK", "ACQUIRE", "RELEASE"

Example:

```
vtk_license_check QUICK
OK
vtk_license_check ACQUIRE
OK
```

Returns:

"OK" or "ERROR"

vtk_licmon

vtk_licmon_get

Usage:

```
vtk_licmon_get
```

Description:

Get the license monitor status in a single compressed message.

vtk_limits

vtk_limits_get

Usage:

```
vtk_limits_get limitArray
```

Description:

Get the list of all limits for the current process. This procedure can be used typically in Tcl based environment definitions. Not available on win64.

Note:

Check the documentation on how the limits can be controlled with the environment variables
VOV_LIMIT_stacksize, VOV_LIMIT_datasize, etc.

Examples:

```
vtk_limits_get array
parray array
array(coredumpsize) = 2147483647
array(cputime)      = 2147483647
array(datasize)     = 2147483647
```

```
array(filesize)      = 2147483647  
array(stacksize)     = 8388608
```

vtk_limits_set

Usage:
vtk_limits_set limitArray

Description:
Set the limits for the current process based on the values of limitArray.
This procedure can be used typically in Tcl based environment definitions.
Not available on win64.

Note:
Check the documentation on how the limits can be controlled with the environment variables
VOV_LIMIT_stacksize, VOV_LIMIT_datasize, etc.

Example:
set array(stacksize) 100000
vtk_limits_set array

vtk_logname

Usage:
vtk_logname

Description:
Get the name of the current user. This works on both Unix and Windows.

Example:
vtk_logname
ftadmin

Returns:
A string

See Also

[vtk_groupname](#)

[vtk_user_get_shell](#)

vtk_lsf

vtk_lsf_parse_resources

Usage:
vtk_lsf_parse_resources [OPTIONS] LSF-Resource-Spec Result-Array

Description:
Parse a LSF resource string and store elements into a tcl array

Example:
vtk_lsf_parse_resources "mem=2333:rusage[dc=1]" result
vtk_lsf_parse_resources -d "mem=2333:rusage[dc=1]" result
parray result
result(order) =
result(resexpr) =
result(select) = mem#2333
result(rusage) = dc#1
result(span,host) = 1
result(span,tile) = 1

Returns:
An NC resource string

vtk_metric

vtk_metric_add

Usage
vtk_metric_add

Description
Add one metric of name NAME in metric family FAMILY(default "DESIGN")
at time TS (default 'now').

vtk_metric_op

Usage
vtk_metric_op

Description
Perform the requested metric operation. Operation is one of: "crop", "save", "load",
"reduce", "reducefamily"

vtk_microcode

vtk_microcode_get

Usage:

```
vtk_microcode_get flags_currently_ignored
```

Description:

Get the multiqueue status in a single compressed message. The argument is currently ignored.

Example

```
vtk_microcode_get 0
```

vtk_microcode_set

Usage:

```
vtk_microcode_set microcode
```

Description:

Set the Altair Allocator status in a single compressed message. The argument is microcode.

Example:

```
vtk_microcode_set microcode
```

vtk_modify_fields

Usage

```
vtk_modify_fields [fieldname1=value1] [fieldname2=value2]...[fieldnameN=valueN]
```

Description

Modifies queryable fields of an object, when possible.

Returns

0 on success, 1 on failure.

vtk_mq_enable_priority_based_alloc

Usage
vtk_mq_enable_priority_based_alloc flag

Description
Enable the project priority-based allocation feature on this License Allocator instance

Example
vtk_mq_enable_priority_based_alloc 1

Returns
Nothing

vtk_mq_get_priority_based_alloc_percentage

Usage
vtk_mq_get_priority_based_alloc_percentage

Description
Get project priority based allocation percentage for this License Allocator instance. Reserved for use by License Allocator daemon that implements user commands.

Returns
Percentage or an error message

vtk_mq_set_priority_based_alloc_percentage

Usage
vtk_mq_set_priority_based_alloc_percentage percentage

Description
Defines the percentage of total available tokens allocated for high-priority projects.
Requires ADMIN privileges and reserved for use by License Allocator daemon that implements user commands.

Example
vtk_mq_set_priority_based_alloc_percentage 10

Returns
Nothing

vtk_mqsite_create_or_modify

ARGS: array

RETURNS: "ok" or an error message

Create or modify a multiqueue site object

Example:

To create a new multiqueue site object:

```
set array(site)           "vnc"
set array(host)           "jaguar"
set array(port)           6271
set array(nickname)       "USA"
set array(version)        "2011.06"
set array(remotecmd)      ""
set array(lsfuser)        ""
set array(lsfdir)         ""
set array(timezone)       ""
set array(localfile)      "/tmp/nc/localfile"
set array(defaultweight)  100
set array(scheduler)      "0"
set array(issl)           "0"
set array(lsfdir)         "0"
set array(enabled)        "0"
set array(projectid)      "1714958736"
vtk_mqsite_create array
```

To modify an existing site object, also set:

```
set array(id)             "103456"
vtk_mqsite_modify array
```

vtk_multiqueue

vtk_multiqueue_get

Usage:

```
vtk_multiqueue_get flags_currently_ignored
```

Description:

Get the multiqueue status in a single compressed message. The argument is currently ignored.

Example:

```
vtk_multiqueue_get 0
```

vtk_multiqueue_get_matches

Usage:

```
vtk_multiqueue_get_matches resourceMap siteName
```

Description:

Get the multiqueue matches status for a given resource and a given site.

Example:

```
vtk_multiqueue_get_matches resourceMap siteName
```

vtk_multiqueue_set

Usage:

```
vtk_multiqueue_set microcode
```

Description:

Use microcode to set lots of resource maps in a single message. Used mostly by vovmultiqueued.

The caller must have ADMIN privileges. The microcode is sent to the remote vovserver in compressed form.

This procedure is efficient but deliberately hard to use. To set a single resource map, use vtk_resourcemap_set instead.

Simple description of the microcode understood by this procedure.

```
v1 BOOL -- Set verbosity.
  The verbosity appears in the
  remote vovserver. log of the
s1 RANK OWNER EXPIRE -- Set Rank Owner
  and Expiration date.
r1 TYPE NAME MAP MAX LMFEATURE -- Set resource map
  parameters.
p1 TYPE NAME PREEMPT_METHOD -- Set preemption
  method for resource map.
H1 start TYPE NAME -- Begin a section
  of handles for a resource map.
h1 USER HOST LICHOST LICPORT CHECKOUT HANDLE TOKENS QUEUEFLAG -- Describe a
  handle for the resource map.
H1 done
J1 start TYPE NAME -- Begin a section
  of jobs using a resource map.
j1 PROJECT HOST USER JOBID STATUS GRABBED REVOKED STARTTS -- Describe a job
  that uses the resource map.
J1 done
```

All other lines are silently ignored.

vtk_new_process_session

Usage

```
vtk_new_process_session
```

Description

Creates a new process session (and group) for the current process. All errors are ignored. No-op on Windows.

vtk_node

vtk_node_barrierinvalid_get

Obsolete

Usage:

```
vtk_node_barrierinvalid_get nodeId
```

Description:

(Obsolete) This procedure is redundant with `vtk_node_get`, which also returns the value of the `barrierInvalid` flag, in addition to many other values.

Returns:

The value of the `barrierInvalid` flag on the node.

vtk_node_cache

Usage:

```
vtk_node_cache [OPTIONS] nodeId setId
```

Description:

Cache a node, given its id, and add it to a specified nodeset. The procedure fails if an illegal `nodeId` or `setId` are passed.

Options:

`-neighbors` Get the node and its neighbors also.

Examples:

```
vtk_node_cache 35173 24123
```

```
vtk_node_cache -neighbors 35173 24123
```

vtk_node_change_status

Usage:

```
vtk_node_change_status nodeId newNodeStatus FLAG WHY_REASON
```

Description:

This procedure changes the status of a node and its children. The argument FLAG is one of:

- * THIS_NODE_ONLY - changes only the specified node
- * ALL_CHILDREN - changes status of all the children
- * NODE_AND_CHILDREN - changes the status of the node and all children
- * AUTOMATIC - the status propagation depends on the status:
 - if status is VALID, only the given node is affected (THIS_NODE_ONLY),
 - if status is INVALID, all nodes are affected (NODE_AND_CHILDREN).

The _FORCE string can be appended to the flags to force status changing for Running and Retracing jobs.

The WHY_REASON is shown in the GUI or other text output that shows why a job has a given status

Examples:

```
vtk_node_change_status 12345 SLEEPING AUTOMATIC "manual invalidation"  
vtk_node_change_status 12345 INVALID NODE_AND_CHILDREN "custom script changed status"
```

Returns:

Nothing

vtk_node_connectivity

Usage:

```
vtk_node_connectivity nodeId
```

Description:

Determine the connectivity of a node.

Example:

```
vtk_node_connectivity 12345  
ISOLATED
```

Returns:

One of ISOLATED INPUT OUTPUT or USED.

vtk_node_forget

Usage:

```
vtk_node_forget nodeId
```

Description:
Forget the given node. If the node is a job, you must be the owner or ADMIN to be able to forget it.

Example:
vtk_node_forget 12345

Returns:
Nothing

vtk_node_format

Usage:
vtk_node_format nodeId format

Description:
This procedure retrieves information about the node based on the format string.

Example:
vtk_node_format 12345 {@STATUS@ @LEVEL@}
VALID 12

@PROPERTIES@ : Returns a list of data type (S=string, I=integer) followed by the property name.

Example:
vtk_node_format 12345 @PROPERTIES@ S FORCE_STATUS_CHANGE S GUI_LABEL

@PROPDUMP@ : Returns A dictionary of property names and property attributes. Each property attribute list consists of a dictionary. The dictionary consists of a description/value pairs.

Example:
vtk_node_format 12345 @PROPERTIES@
FORCE_STATUS_CHANGE { type S value {set status to VALID by clif} }
GUI_LABEL { type S value {shortLabel} }

vtk_node_get

Usage:
vtk_node_get [-usecache] nodeId [array]

Description:
Returns information about a node (job or file). If the nodeId doesn't exist, an error is returned. With just the nodeId a list of five elements is returned:

- * node status
- * BARRIER or NORMAL
- * timestamp
- * FILE | TOOL

```
* node name -- NO LONGER MEANINGFUL!  
* node extended status
```

With the optional array argument, the array is filled with information about the node.

Examples:

With the placeId as unique argument:

```
vtk_node_get 1234565  
INVALID NORMAL {Sat Apr 5 09:45:20 2003} TOOL no_name SCHEDULED
```

With the optional array argument:

```
vtk_node_get 3137 nodeInfo  
nodeInfo(status) = INVALID  
nodeInfo(type) = NORMAL  
nodeInfo(timestamp) = 1025278326  
nodeInfo(objtype) = TOOL  
nodeInfo(extendedstatus) = SCHEDULED
```

Returns:

A list of 6 elements

vtk_node_get_ios

Usage:

```
vtk_node_get_ios nodeId formatString
```

Description:

This procedure retrieves the inputs and outputs of a node.

Example (result modified for clarity):

```
vtk_node_get_ios 12345 "@ID@ @STATUS@ @STICKY@"
```

```
{  
  {04395184 VALID N} {02049739 VALID N} {02049881 VALID S} {02050021 VALID N}  
} {  
  {02055399 INVALID N}  
}
```

Returns:

```
{list of inputs} {list of outputs}
```

vtk_node_get_sets

Usage:

```
vtk_node_get_sets nodeId or setId
```

Description:

Get the list of all sets to which the node or set belongs.

Example:

```
vtk_node_get_sets 12345
```

Returns:

A list of {id name} pairs for all sets the node or set belongs to.

vtk_node_history

Usage:

```
vtk_node_history nodeId FLAGS direction levels
```

Description:

This procedure returns a formatted string representing a depth-first traversal of the trace starting from the nodeId as specified by direction and levels. Direction is either "FORWARD" or "BACKWARD". FLAGS is a list of one or more of the following tokens: ANNOTATION PLACE TRANSITION NODE.

Returns:

A formatted string

vtk_node_impact

Usage:

```
vtk_node_impact nodeId [array]
```

Description:

This procedure returns a summary of jobs and files in the downcone of the given node.

With one argument, the procedure returns a formatted string, such as the one shown below.

This usage is discouraged. With two arguments, the procedure fills the relevant fields in the specified array (also shown below in the example).

Examples:

```
vtk_node_impact 12345
VALID NODES Files: 22 Tools: 5 Duration:3m15s
OTHER NODES Files: 21 Tools: 17 Duration:15m21s
-----
```

```
TOTAL Files: 43 Tools: 22 Duration:18m36s
```

```
vtk_node_impact 4567 a
```

```
parray a
```

```
a(other,duration)    = 181
```

```
a(other,files)       = 10
```

```
a(other,jobs)        = 7
```

```
a(other,unknown)     = 0
```

```
a(valid,duration)    = 1092
```

```
a(valid,files)       = 223
```

```
a(valid,jobs)        = 180
```



```
a(valid,unknown)      = 0
```

Returns:

A formatted string or "ok|error"

vtk_node_status_logging

Usage:

```
vtk_node_status_logging nodeOrSetId  OPTIONAL_LOGGING_TYPE  -recursive
```

Description:

This procedure sets or gets the WHYSTATUS reason logging type for a node or a set (applies to all nodes in the set). Within a set, the recursive option applies to nodes in subsets recursively. If the ID of System:nodes is provided, the change applies to all nodes. This can be used to globally enable or disable status logging. If no logging type is provided, the logging type of nodeOrSetId is returned. The log entries are recorded in the server journals, not the server log. Getting the value for a set returns a summary in the format "UNSET v NONE w ALL x INVALID y DOWNCONE z TOTAL t". If the ID is 0, set the trace policy setting which acts as a global on all nodes. The node's value overrides it unless UNSET. Logging type is one of:

- * UNSET; Use the trace policy setting (default UNSET)
- * NONE; Disables logging of status changes
- * ALL; Enables logging of all status changes
- * INVALID; Enables logging of status turning INVALID
- * DOWNCONE; Enables logging of INVALID status propagating to downcone nodes

Examples: (12345 is a job or file ID, 678 is a set ID)

```
vtk_node_status_logging 12345
vtk_node_status_logging 12345 ALL
vtk_node_status_logging 678
vtk_node_status_logging 678 -recursive
vtk_node_status_logging 678 DOWNCONE
vtk_node_status_logging 678 NONE -recursive
vtk_node_status_logging [vtk_set_find System:nodes] NONE
vtk_node_status_logging 0 ALL
vtk_node_status_logging 0 NONE
```

Returns:

If getting value, the logging type of the node. If setting value, the number of nodes that changed logging type

vtk_node_why

Usage:

```
vtk_node_why nodeId maxReasons
```

Description:

This procedure retrieves the reasons for the status of the specified node. The argument maxReasons is silently limited in the range [1,100].

Example:

```
vtk_node_why 12345 4
clevercopy -m 644 ./OBJ/FTaux/FTaux.pdf ../../../../bui...
is INVALID
Invalid input: 02049603 ${BUILD_TOP}/src/html/FTaux.pdf
1. (FILE) ${BUILD_TOP}/src/traceui/vtknode.cc
is younger by 2m13s
NodeId=00003475
```

Returns:

A formatted string.

vtk_nodeeditor

Usage:

```
vtk_nodeeditor Node Editor browser widget
```

Description:

This procedure manage the node editor in the VOV console. It is designed to be used by the GUI developer, not by the regular user.

Example:

```
vtk_nodeeditor <widget>
vtk_nodeeditor <widget> update
vtk_nodeeditor <widget> setid ID
```

vtk_nodes

vtk_nodes_disconnect

Usage:

```
vtk_nodes_disconnect firstNodeId secondNodeId
```

Description:

Disconnect the two nodes, if they are connected. The order of the two ids is not important.

Returns:

Nothing

See Also

[vtk_input_declare](#)

[vtk_output_declare](#)

vtk_nodestatus

vtk_nodestatus_get_color

Usage:
vtk_nodestatus_get_color nodeStatus

Description:
This procedure get the color corresponding to the given status in the format
"#xxxxxx"

Examples:
vtk_nodestatus_get_color VALID
#70B070
vtk_nodestatus_get_color FAILED
#FF0000

Returns:
A string.

vtk_nodestatus_get_mask

Usage:
vtk_nodestatus_get_mask nodeStatus1 ...

Description:
This procedure generates mask values to be used with the @STATUSMASK@ field.

Examples:
vtk_nodestatus_get_mask VALID
1
vtk_nodestatus_get_mask FAILED INVALID
130

Returns:
An integer

vtk_nodestatus_get_ncname

Usage:
vtk_nodestatus_get_ncname nodeStatus isScheduledFlag isSuspendedFlag

Description:
This procedure generate the name of the status for network computing.

Examples:
vtk_nodestatus_get_ncname SCHEDULED
Queued
vtk_nodestatus_get_ncname RUNNING 0 1
SuspendedRT
vtk_nodestatus_get_ncname INVALID 0 0 1
OnDemand

Returns:
An integer

vtk_nodestatus_set_color

Usage:
vtk_nodestatus_set_color nodeStatus bgColor fgColor

Description:
Set a new color for nodestatus. To change colors of transitions, use
vtk_transition_color_control

Examples:
vtk_nodestatus_set_color VALID blue white

Returns:
A string.

vtk_nodeviewer

Usage:
vtk_nodeviewer Node Viewer browser widget

Description:
This procedure is deprecated. Please use vtk_nodeeditor instead.

See Also

[vtk_nodeeditor](#)

vtk_object

vtk_object_get

Usage:
vtk_object_get objectId <array>

Description:
This procedure is used to find the type of an object starting from the id.

Example:
vtk_object_get 2
a(type) = NODESET

Returns:
Nothing

vtk_object_get_fromcache

Usage:
vtk_object_get_fromcache objectId <array>

Description:
This procedure is used to find the type of an object starting from the id.

Example:
vtk_object_get_fromcache 2
a(type) = NODESET

Returns:
Nothing

vtk_output

vtk_output_declare

Usage:
vtk_output_declare transId placeId [flags]

Description:
Without flags, declare an output with force, meaning that output conflicts are resolved with force.
The flags can be:

- * shared, if the output is shared among many {jobs;} all jobs must declare the output as shared;

- * force, if output conflicts are to be ignored (this is the default);
- * noforce, if output conflicts are to generate an error;
- * stop_barrier, if the output has a barrier, and the barrier is effective to stop insignificant changes;
- * propagate_barrier, if the output has a barrier, and the barrier allows changes to propagate to the output;
- * ignore_timestamp, if the timestamp of the output is not related to the job execution time
- * optional, if the output may not exist, in which case the dependency is dropped upon completion of the job.
- * sticky, if the output has to remain attached to the transition even if the dependency ceases.
- * fileready, used by a running job to indicate that the output is available for processing by its downstream.
In other words, there will be no further changes to this output even though the job is still running.
- * log, the output is one of the main logs of the job
- * normal, an option with no effect

Examples:

```
vtk_output_declare 12345 4567  
vtk_output_declare 12345 4567 -ignore_timestamp
```

vtk_output_declare_ex

Usage:

```
vtk_output_declare_ex transId placeName [flags]
```

Description:

An alternative to `vtk_output_declare`, which uses 2 VovIds, while this uses the name of the output.

Without flags, declare an output with force, meaning that output conflicts are resolved with force.

The flags can be:

- * shared, if the output is shared among many jobs; all jobs must declare the output as shared;
- * force, if output conflicts are to be ignored (this is the {default});
- * noforce, if output conflicts are to generate an error;
- * stop_barrier, if the output has a barrier, and the barrier is effective to stop insignificant changes;
- * propagate_barrier, if the output has a barrier, and the barrier allows changes to propagate to the output;
- * ignore_timestamp, if the timestamp of the output is not related to the job execution time
- * optional, if the output may not exist, in which case the dependency is dropped upon completion of the job.
- * sticky, if the output has to remain attached to the transition even if the dependency ceases.
- * log, the output is one of the main logs of the job
- * nocheck, avoid checking the file on the server side
- * normal, resets all previous options
- * db DB

Examples:

```
vtk_output_declare_ex 12345 output.log  
vtk_output_declare_ex 12345 output.log -ignore_timestamp
```

See Also

[7.61vtk_nodes](#)

vtk_path

vtk_path_canonicalize

Usage:

```
vtk_path_canonicalize path [-ignorelinks]
```

Description:

This procedure computes the canonical path of a file. A canonical path is one without symbolic links, without "dots", and, when possible, in logical form, as defined by the equivalence rules. With the "-ignorelinks" option, it does not expand/follow symlinks.

Returns The canonical version of the path.

Examples

```
vtk_path_canonicalize .  
${TOP}/units/abc
```

See Also

[7.18vtk_equivalence](#)

See Also

[vtk_equivalence](#)

vtk_path_expand

Usage:

```
vtk_path_expand path [-ignorelinks]
```

Description:

This procedure returns the expanded version of a path, i.e. a path where the logical name has been expanded. With the -ignorelinks option, it does not expand/follow symlinks.

Examples:

```
vtk_path_expand .  
/export/projects/chip1/units/abc
```

Returns:

The expanded version of the path.

See Also

[7.18vtk_equivalence](#)

vtk_path_flush_caches

No information available.

vtk_path_flush_nfs_cache

No information available.

vtk_path_get_relative

Usage:
vtk_path_get_relative path

Description:
Old name, for backwards compatibility. Please use vtk_path_relativize instead.

vtk_path_is_excluded

Usage:
vtk_path_is_excluded path

Description:
This procedure tests whether the path is excluded by the exclusion rules. The path must be passed in canonical form. The old name 'vtk_path_exclude' is also supported for backwards compatibility.

Examples:
vtk_path_is_excluded \$env(VOVDIR)/etc
0
vtk_path_is_excluded [vtk_path_canonicalize \$env(VOVDIR)/etc]
1

Returns:
1 if the file is to be excluded, 0 otherwise.

See Also

[vtk_exclude_rule](#)

vtk_path_mk_symlink

Usage:
vtk_path_mk_symlink path symbolic

Description:
This procedure make symbolic link

Example:
vtk_path_mk_symlink ../mypath/dir1/dir2 ../linkTmdir2

Returns:
TCL_OK

vtk_path_permissions

No information available.

vtk_path_print_caches

No information available.

vtk_path_realpath

Usage:
vtk_path_realpath path

Description:
This procedure returns the full expanded version of a path, with all symlinks resolved. In other words, the full real path.

Examples:
vtk_path_realpath .
/export/projects/chip1/units/abc

Returns:
The expanded full real version of the path.

See Also

[7.18vtk_equivalence](#)

vtk_path_relativize

Usage:

```
vtk_path_relativize path
```

Description:

Return the relative name of the given full path with respect to the current directory.

This procedure is useful to compute portable command lines, with relative paths instead of hard-coded full paths.

This procedure is named similarly to `vtk_path_canonicalize`. The relative path will have a number of `..` components up to a maximum specified by argument `maxdotdot` which has a default value of 5.

vtk_percent

No information available.

vtk_pie

vtk_pie_chart

Usage:

```
vtk_pie_chart [OPTIONS] data1 ...
```

Description

Generate a GIF file representing a pie chart. The data can be in the form `NUMBER` or `NUMBER:LABEL`.

If no label is shown, then the empty label is used.

- * `-b`: Border around image (>1) (10 pixels). If the border is greater than 50, then the labels are also shown
- * `-w`: Width of image (400 pixels)
- * `-h`: Height of image (400 pixels)
- * `-p`: Print label% data inside slices
- * `-l`: Print label data inside slices
- * `-L`: Print label index inside slices
- * `-s`: Limit Display data: to (si) data items, sum remainder in data[si+1]
- * `-t`: Output a Tcl script to map entries to colors
- * `-f`: Name of output GIF file (\"out.gif\")

Examples:

```
vtk_pie_chart -w 800 -h 400 -f mygraph.gif 10 20 30 44 55
```

```
vtk_pie_chart -w 800 -h 400 -f mygraph.gif 10:us 20:eu 30:ch 44:jp 55:ru
```

vtk_ping

Usage:
vtk_ping OPTIONAL_NUMBER_OF_PINGS

Description:
Ping the vovserver. Wait for reply. The default is 1. The max is 1000.

Returns:
The time it takes to perform the pings, in seconds

vtk_place

vtk_place_find

Usage:
vtk_place_find database filename [-nocheck]

Description:
Find a place with given database and name. If the option -nocheck is given, then the timestamp of the place is NOT checked.

Example:
vtk_place_find FILE /usr/bin/cp
00012345
vtk_place_find FILE /usr/bin/cp -nocheck
00012345

Returns:
The id of the place or 0 if the place could not be found.

vtk_place_get

Usage:
vtk_place_get placeId [array]

Description:
We recommend that you always pass the second argument, which is an array, because this is the new syntax and yields much more detail.
The old syntax, with just the placeId, returns a list of:

1. status
2. barrier_flag
3. node_timestamp

```
4. db
5. name
6. file_timestamp
```

With the optional array argument, the array is filled with information about the place.

Examples:

With the placeId as unique argument:

```
vtk_place_get 3137
VALID BARRIER {date} FILE ${TOP}/placeinfo.hh {date}
```

With the optional array argument:

```
vtk_place_get 3137 fileInfo
parray fileInfo fileInfo(db)      = FILE
fileInfo(dir)                     = ${BUILD_TOP}/include
fileInfo(name)                    = ${BUILD_TOP}/include/placeinfo.hh
fileInfo(status)                  = VALID
fileInfo(tail)                    = placeinfo.hh
fileInfo(timestamp)               = 1025278326
fileInfo(md5flag)                 = 0
```

Returns:

A list

vtk_place_get_or_create

Usage:

```
vtk_place_get_or_create database, placeName, [-zippable | -check | -canonical | -
recanonicalize | -log | -nop | -trigger ACTION | -md5 boolean ] database, placeName,
[0|1]
```

Description:

Also called `vtk_place_find_or_create`. Find or create a place by providing database and name.

The optional third argument 'checkIt' determines whether the server should check the place

(generally a file with `stat()`) or not. This has implication with NFS cacheing. The default value

for the flag is 0, meaning that the server does not check the file.

Options:

* -zippable	The place is zippable.
* -check	The place timestamp will be checked by the server upon creation.
* -canonical	The name of the place is already canonical.
* -recanonicalize	The name may need to be recanonicalized.
* -trigger,run	Trigger a retrace of the downcone.
* -trigger,stop	Trigger a stop of all running jobs of the downcone.
* -md5 boolean	Use md5 for barriers.
* -log	Label this place as an "important log"
* -nop	Do nothing.

Example:

```
vtk_place_get_or_create "FILE" abc
vtk_place_get_or_create "FILE" $TOP/data/abc -canonical
vtk_place_get_or_create "FILE" $TOP/data/abc -md5 1
```

```
vtk_place_get_or_create "FILE" $TOP/data/main.log -log
```

Return:
The id of the place.

vtk_place_list_find

Usage:
vtk_place_list_find format, listOfNames

Description:
Given a list of names, find all the corresponding files, reporting information using the given format.
If the file is not found in the trace, the string UNKNOWN is reported. This procedure is used, for example, by the command vls.

Examples:
set listOfNames [glob [pwd]/*.html]
vtk_place_list_find "@ID@ @STATUS@ @TAIL@" \$listOfNames
02049511 VALID vncsummary.html
00014103 VALID vncgroups.html
00014099 VALID vncinstall.html
UNKNOWN zx.html

vtk_place_set

Usage:
vtk_place_set placeId array

Description:
Only the fields db, name, and zippable can be set

Example
set array(name) "abc"
set array(db) "FILE"
set array(barrier) "1"
vtk_place_set \$placeId array

vtk_place_update

Usage:
vtk_place_update placeId

Description:

The main reason for this procedure to exist is to allow a client to update `TIMESTAMP` information about a file in cases in which the vovserver may be unable to get the correct information because of the NFS cache.

This procedure will update the following information: existence, timestamp, zipped flag. This procedure works only for places in the DB "FILE", those most affected by the NFS caching problem. All other places are silently ignored.

Example:

```
set placeId [vtk_place_find "FILE" $fileName -nocheck]
vtk_place_update $placeId
```

vtk_port

vtk_port_number

Usage:

```
vtk_port_number [-h] [-checkenv] [-envname ENVNAME] [-client/-server] [project_name]
```

Description:

This procedure returns the port used by the server. If the `project_name` argument is not passed, then the environment variable `VOV_PROJECT_NAME` is used.

Examples:

```
vtk_port_number
6271
vtk_port_number licadm
6306
vtk_port_number -server -envname VOV_PORT_NUMBER licadm
```

Returns:

The port number for given `project_name`

vtk_ppid

No information for this procedure.

vtk_preemptrule

vtk_preemptrule_create

Usage:

```
vtk_preemptrule_create array
```

Description:

The array contains the information of the preemption rule creation, including:

- * pool
- * rulename
- * ruletype
- * enabled
- * debug
- * preempting
- * waitingfor
- * bucketag
- * multiqueueres
- * mqthresh
- * donotdisturb
- * preemptable
- * preempttaskerspec
- * preempttaskernum
- * killage
- * method
- * skippresumedjob
- * reservetime
- * reservetype
- * reservenum
- * reservetasker
- * reservefor
- * resumeres
- * resumedelay
- * numjobs
- * maxattempts
- * sortjobsby
- * script
- * id
- * order
- * fireonce

For detailed explanation for each field, refer to Preemption Rules

Returns:

The preemption rule Id of the preemption rule just created, or an error message.

vtk_preemptrule_delete

Usage:

```
vtk_preemptrule_delete preemptRuleId
```

Description:

This procedure is used to delete a preemption rule object from server.

Example

```
vtk_preemptrule_delete 00358286  
ok
```

Returns:

"ok" or an error message

vtk_preemptrule_delete_all

Usage:
vtk_preemptrule_delete_all

Description:
This procedure is used to delete all preemption rule objects from server.

Example:
vtk_preemptrule_delete_all

Returns:
"ok"

vtk_preemptrule_find

No information is available.

vtk_preemptrule_get

Usage:
vtk_preemptrule_get preemtRuleId array

Description:
This procedure fills the array with the value of the preemption rule.

Examples:
vtk_preemptrule_get 189823 value
parray value
value(bucketage) = 4
value(count,firings) = 0
value(debug) = 0
value(donotdisturb) = 600
value(enabled) = 1
value(errormsg) =
value(fireonce) = 0
value(killage) = 0
value(maxattempts) = 0
value(method) = SUSPEND
value(mqthresh) = 0.900000
value(multiqueueres) =
value(numjobs) = 1
value(order) = 22
value(pool) = ReprTestPool
value(preemptable) = jobclass==@JOBCLASS@ priority<@PRIORITY@
value(preempting) = jobclass==testmapp priority>=5
value(preempttaskernum) = 1
value(preempttaskerspec) =
value(reservefor) =


```
value(reservenumber)      = 0
value(reservetasker)      =
value(reservetime)        = 0
value(reservetype)         = JOBID
value(resumedelay)        = 5
value(resumeres)          =
value(ruleName)            = RegrTestJOBCLASS1523045905
value(ruleType)            = GENERIC
value(script)              =
value(skippresumedjob)    = 0
value(sortjobsby)         =
value(ts,last,active)      = 0
value(ts,last,fired)       = 0
value(waitingfor)          = License:regmappedres
```

Returns:
"ok" or an error message

vtk_preemptrule_modify

Usage:
vtk_preemptrule_modify array

Description:
The array contains the information of the preemption rule modification, including:

- * pool
- * ruleName
- * ruleType
- * enabled
- * debug
- * preempting
- * waitingfor
- * bucketag
- * multiqueueres
- * mqthresh
- * donotdisturb
- * preemptable
- * preempttaskerspec
- * preempttaskernum
- * killage
- * method
- * skippresumedjob
- * reservetime
- * reservetype
- * reservenumber
- * reservetasker
- * reservefor
- * resumeres
- * resumedelay
- * numjobs
- * maxattempts
- * sortjobsby
- * script
- * id
- * order
- * fireonce

For detailed explanation for each field, refer to Preemption Rules

Returns:

The preemption rule Id of the preemption rule, or an error message.

vtk_product

vtk_product_get_info

Usage:

```
vtk_product_get_info option
```

Description:

Return the name, version, or copyright statement of the current product.
The option argument is one of: -version -copyright -name.

Examples:

```
vtk_product_get_info -version  
2021.1  
vtk_product_get_info -copyright  
Copyright (c) 1995-2021, Altair Engineering  
vtk_product_get_info -name  
ft
```

Returns:

Information about the current product.

vtk_product_name

No information is available.

vtk_prop

vtk_prop_decr_and_get

Usage:

```
vtk_prop_incr_and_get objectId propName
```

Description:

This procedure decrements and returns the value of the property "propName"

Returns:

The integer or string value of the property, or raises an error if the property is not defined for the object. Value cannot be decremented below 0.

vtk_prop_delete

Usage:

```
vtk_prop_delete objectId propName
```

Description:

This procedure deletes the given property from the given object.

Examples:

```
vtk_prop_delete 2 MY_INTEGER_PROPERTY
```

Returns:

An error if the property was not attached to the object.

vtk_prop_get

Usage:

```
vtk_prop_get objectId propName
```

Description:

This procedure returns the value of the property "propName" of the given object.

Returns:

The integer or string value of the property, or raises an error if the property is not defined for the object.

vtk_prop_incr_and_get

Usage:

```
vtk_prop_incr_and_get objectId propName
```

Description:

This procedure increments and returns the value of the property "propName"

Returns:

The integer or string value of the property, or raises an error if the property is not defined for the object.

vtk_prop_list

Usage:

```
vtk_prop_list objectId
```

Description:

Get the name of all properties attached to the specified object.

Returns:

A list of names, or an error if the object does not exist.

vtk_prop_set

Usage:

```
vtk_prop_set [OPTIONS] objectId propName propValue
```

Description:

This procedure sets the value of a property attached to an object. The -auto/-text/-integer options are used to specify the property data type.

The default behavior (-auto) is to automatically determine the data type by looking at the first character of the specified property value.

The -asynch option is used to prevent the client from waiting for confirmation from the server, which results in slightly faster execution.

The default behavior is to wait (-synch).

The -elements option is used to set the property on all elements of a set, if the specified object ID is that of a set.

The default behavior is to apply the property to the set itself (-noelements).

The -field option is used internally by preemption only and directs the specified value into a field on the object instead of a property.

Example:

```
vtk_prop_set 2 MY_INTEGER_PROPERTY 12
vtk_prop_set -text 2 MY_TEXT_PROPERTY "Hello there"
vtk_prop_set -asynch -text 2 MY_TEXT_PROPERTY "Hello there"
vtk_prop_set -elements -text 2 MY_TEXT_PROPERTY "Hello there"
```

Returns:

Nothing

vtk_protocol

vtk_protocol_id

Usage:

`vtk_protocol_id`

Description:

This procedure returns the negotiated protocol id between this client and the vovserver.
The protocol id is an integer.

Examples:

```
vtk_protocol_id  
20101116
```

Returns:

An integer representing the protocol id.

vtk_protocol_info

Usage:

```
vtk_protocol_info [-id | -capability | -stat resultArray | -nid | -ncapability | -  
nstat resultArray ] );
```

Description:

This procedure returns the information about the current protocol between this client and the vovserver by default, The protocol id is an integer.

Example:

```
vtk_protocol_info 20101116
```

Returns:

An integer representing the protocol id.

vtk_pty_server_open

Usage:

```
vtk_pty_server_open [OPTIONS]
```

Options:

```
* -sig          Process signals (ctrl-C and ctrl-Z) locally  
                By default, ctrl-C and ctrl-Z are passed to  
                the remote client.  
  
* -splitstderr  Don't combine the job's stdout and stderr output.  
  
-logfile PATH  Path to logfile
```

vtk_reservation

vtk_reservation_create

Usage:

```
vtk_reservation_create type what quantity start end [Options]
```

Description

Creates a reservation for tasker, resourcemap or emulator. A reservation with type tasker is the same with a reservation created using vtk_tasker_reserve. A reservation with type resourcemap is the same with a reservation created using as vtk_resourcemap_reserve. type is one of tasker, resourcemap, and emulator.

For tasker reservations, what is the name of tasker. If the reservation is for multiple taskers, list them comma separated. quantity is not used.

For resourcemap reservations, what is type:name of the resource. quantity can be the number of tokens if the resource is a license. start is the start time of the reservation. end is the end time of the reservation. The value of end can be forever if the reservation should never end.

Option	Description
-user username	Reserve for specified user or users
-group groupname	Reserve for specified fairshare group or groups
-usergroup groupname	Reserve for specified user group or groups
-osgroup groupname	Reserve for specified Unix (or OS) group or groups
-jobclass jobclass	Reserve for specified jobclass or jobclasses
-jobproj jobproj	Reserve for specified jobproj
-bucketid bucketid_list	Reserve for specified bucket id or ids
-id id_list	Reserve for specified id or ids
-resources "SLOTS/x RAM/x CORES/x SWAP/x specified tasker"	Reserve given number of resources of specified tasker
-update	Update an existing reservation if all other parameters are the same except start time and end time.

Example:

```
vtk_reservation_create tasker 'host1,host2' 4 1484612764 1484623650 -user brian
vtk_reservation_create type what quantity start end -user
  user1,user2
vtk_reservation_create tasker localhost 1 1484612764 1484623650 -user
  john
vtk_reservation_create tasker localhost 1 1484612764 1484623650 -
group g1,g2
vtk_reservation_create resourcemap License:abc 4 0 forever -
jobclass class1
vtk_reservation_create emulator PAL1 4 1484612764 1484623650 -
group g1 -jobproj chipA -jobclass class1
```

Returns:

VOV id of created reservation. 'nochange' if the reservation already exists.

Note:

Two reservations are considered the same if all parameters to create the reservations are the same.

vtk_reservation_delete

Usage:
vtk_reservation_delete id

Description:
Deletes the reservation having VOV id 'id'.

Returns:
ok if found, or errors

vtk_reservation_delete_orphans

Usage:
vtk_reservation_delete_orphans

Description:
Deletes all orphan reservations.

Returns:
ok if no error, or errors

vtk_reservation_get

Usage:
vtk_reservation_get id arrayname

Description:
Retrieves information into the array named by 'arrayname' about the reservation

Returns:
ok if found, or errors if there is no such reservation

vtk_reservation_update

Usage:
vtk_reservation_update id fieldname value

Description:
Modify reservation contents. Available fieldnames can be found using vovselect fieldname from reservations.

Returns:
ok if update is successful, or errors

vtk_resource

vtk_resource_delete

Usage:
vtk_resource_delete resourceName

Description:
Delete the named resource from the resources map.

Note:
Use vtk_resourcemap_del instead.

See Also

[vtk_resourcemap_delete](#)

vtk_resourcemap

vtk_resourcemap_add_lm_handles

Usage:
vtk_resourcemap_add_lm_handles array

Description:
Used by vovresourced and vovlad. Not intended for end user. The array contains the information of the flexlm handles (jobs) to be added, including:

- * array(resource) -- The name of the resourcemap
- * array(count) -- The number of handles to be described
- * array(%d,host)
- * array(%d,user)
- * array(%d,pid)
- * array(%d,licenseHost)
- * array(%d,licensePort)
- * array(%d,lastCheckOut)
- * array(%d,handle)
- * array(%d,tokens)
- * array(%d,queued)

Returns:

"OK" if added successfully, or an error message.

vtk_resourcemap_add_nc_jobs

Usage:

```
vtk_resourcemap_add_nc_jobs array
```

Description:

Used by vovresourced and possibly vovlad. Not intended for end user. The array contains the information of the flexlm handles (jobs) to be added, including:

- * array(resource) -- The name of the resourcemap
- * array(count) -- The number of handles to be described
- * array(%d,project) --
- * array(%d,jobStatus) --
- * array(%d,jobId) --
- * array(%d,pidList) -- List of PIDs associated with this job
- * array(%d,host) -- Execution host of job
- * array(%d,user) -- User of job
- * array(%d,startDate) -- Start timestamp of job
- * array(%d,grabbed) -- Number of tokens grabbed
- * array(%d,revoked) -- Number of tokens revoked by preemption

Returns:

"OK" if added successfully, or an error message.

vtk_resourcemap_change_grab

Usage:

```
vtk_resourcemap_change_grab jobId resourceName quantity
```

Description:

Used only by ADMIN or by the OWNER of the job. If quantity is positive, the resource is forcefully grabbed. If the quantity is negative, then we see if a resource can be released.

If the quantity is zero, the whole routine is still executed but an error will be generated.

Returns:

Nothing.

vtk_resourcemap_clear_reservations

Usage:

```
vtk_resourcemap_clear_reservations resource_name
```

Description:

Example:

```
vtk_resourcemap_clear_reservations License:abc
```

Returns:

Nothing

vtk_resourcemap_delete

Usage:

```
vtk_resourcemap_delete resourceName
```

Description:

Delete the given resource map associated with the given resource. This procedure operates by name. Please use `vtk_resourcemap_forget` to delete a resource by id.

Example:

```
vtk_resourcemap_delete Queue:regression
```

Returns:

Nothing

vtk_resourcemap_forget

Usage:

```
vtk_resourcemap_forget resourceId
```

Description:

Delete the resource map identified by ID `resourceId`. If the resource map is assigned to any currently job, the deletion will be deferred until any such jobs complete.

Note:

The `vtk_resourcemap_delete` function performs the same deletion on a resource map identified by name.

Example:

```
vtk_resourcemap_forget 000001053
```

Returns:

Nothing

vtk_resourcemap_get

Usage:

```
vtk_resourcemap_get nameOrId array
```

Description:

This procedure retrieves information about a resource map. The procedure works more like `vtk_resourcemap_find`, in the sense that it errors out if the resource is not found, and it returns the `VovId` of the resourcemap if the resource is found. To get all matches info, use the name and not the `vovid`.

Example:

```
vtk_resourcemap_get Priority:normal resInfo
parray resInfo
resInfo(available)           = 4
resInfo(expiration)          = 2147483647
resInfo(found)               = 1
resInfo(inuse)               = 0
resInfo(lastevent)           = 1049501790
resInfo(lastuse)              = 1049501785
resInfo(lefttoexpire)        = 1097981857
resInfo(map)                  =
resInfo(max)                  = 4
resInfo(name)                 = Priority:normal
resInfo(notcompathistory)     =
resInfo(others)               = 0
resInfo(owner)                = casotto@bison
resInfo(reserved)             = 0
resInfo(utilization)          = 18.3783
```

...and possibly other fields...

vtk_resourcemap_list

Usage:

```
vtk_resourcemap_list rx
```

Description:

Return the list of resources that match the given regexp.

vtk_resourcemap_reserve

Usage:

```
vtk_resourcemap_reserve resource_name TYPE who howmany duration explanation
```

Description:

Where TYPE is one of USER, GROUP, JOBCLASS, JOBPROJ, JOBID

'duration' may optionally be forever to indicate an expiration that never expires.

Examples:

```
vtk_resourcemap_reserve License:abc USER      john          1 3m ""
vtk_resourcemap_reserve License:abc GROUP    /time/users    3 1h ""
vtk_resourcemap_reserve License:abc JOBCLASS hsim          3 1h ""
vtk_resourcemap_reserve License:abc JOBPROJ  ChipA         20 3w ""
vtk_resourcemap_reserve License:abc JOBID    00012345       1 2h "Preemption"
```

Returns:

Nothing

vtk_resourcemap_return

Usage:

```
vtk_resourcemap_return jobId resourceName [resumer_JobId]
```

Description:

Return all the resources (stolen with `vtk_resourcemap_steal`) to the job from the resumer job if it is specified. If the job has stolen resource, then all stolen tokens are returned to the job. If the job has no stolen resources, nothing happens. This procedure is normally used only by the preemption daemon upon resumption of a suspended job.

Examples:

```
vtk_resourcemap_return 0123456 License:sim
vtk_resourcemap_return 0123456 License:sim 0234567
```

Returns:

Nothing.

See Also

[vtk_resourcemap_steal](#)

[vtk_resourcemap_change_grab](#)

vtk_resourcemap_set

Usage:

```
vtk_resourcemap_set name [OPTIONS]
```

Description:

This procedure is used by an ADMIN user to set a resource map. It is normally used by `vovresourced` in the `resources.tcl` file in the server configuration directory. It is also possibly used by `vovlad.tcl`. The argument name is the name of the resource map, which is of the form 'type:name' or just 'name'.

Options:

```
* -total N Specify the total number available for the resources. The argument N must
  be non-negative can also be "UNLIMITED" (case-insensitive)
* -max N Same as -total. Please prefer -total
* -lmfeature <featureSpec> The specification for the feature (from LicenseMonitor)
  associated with the resource map. The format for featureSpec is "FEATURE_NAME
  PORT@HOST:DAEMON_NAME". Example: "hsim 6500@lxlic02:snsdpd:
* -map <resexpr> A resources expression to be used if the resource map is selected
  during scheduling of a job. The default is the empty string, meaning no map.
* -expire <TimeSpec> After the specified TimeSpec, the resource map expires and is
  deleted if it is not used. After expiration, another agent can take control of the
  resource map. The default value is "never" or "", meaning no expiration.
* -owner <owner> This is the name of the owner of the resource map. Typically, this
  is either "vovresourced:USER@HOST:PORT" or "vovlad:USER@HOST:PORT". This is used
  mostly for documenting who is controlling the resource map.
* -log, -nolog Tell the system to not create a log entry every time the resource map
  changes.
* -ooq, -noooq Tell the system to not compute the "out-of-queue" handles for the
  resource map. This only applies for resource maps with rank less than 20.
* -match, -nomatch Tell the system to not compute any matches for this resource.
* -recent, -norecent When computing the "out-of-queue" handles, include all jobs that
  have recently completed in the matching. This is useful for features that are used
  for short periods of time
* -rank <N> Used to distinguish different controlling agents for the same resource
  map. Normally the rank for vovresourced is 3 and the rank for vovlad is 20. If
  multiple agents compete to control the same resource, the agent with higher rank
  prevails. This is used if multiple redundant vovlad daemons are in use. If the rank
  is 20 or greater, the out-of-queue computation is also disabled.
* -async <0|1> Control whether the communication with vovserver is asynchronous.
  If the value is 1, then the calling agent does not wait for an answer from the
  vovserver. By default, the protocol is synchronous. The asynchronous mode is useful
  to speedup the setting of hundreds of resources. This is now (2011.09) superceded by
  the more efficient "vtk_multiqueue_set" procedure.
* -clearjobs <0|1> Clear all matching information for the resource map.
* -updatelimits Expand @KEYWORD@ to find matching resources to update with new total.
* -limitexceptions <LIST> List of "name total" pairs to be processed differently for
  limit keyword expansion.
* -local, -nolocal Specifies that this simple limit resource is to be managed locally
  in a vovwxd environment
* -preempt <METHOD> (Obsolete: do not use)
```

Old Syntax:

The argument max is either an integer or the string "UNLIMITED". The argument map is a resource expression, i.e. it could be an empty string, the name of another resource, a list of resources, or a combination of resources containing the OR '|' symbol.

Examples (current syntax):

```
vtk_resourcemap_set N5 -total 5 -nolog
vtk_resourcemap_set Lic:a -lmfeature a -total $N -expire 2m
vtk_resourcemap_set Lic:a -lmfeature a -total $N -expire 2m -rank 5
vtk_resourcemap_set Lic:a -lmfeature a -total $N -expire 2m -owner la -rank 5
vtk_resourcemap_set Lic:a -lmfeature a -total $N -expire 2m -owner rd -rank 3
vtk_resourcemap_set Lic:a_@USER@ -total 4 -updatelimits -limitexceptions { Lic:a_joe
  2 Lic:a_bob 6 }
```

Examples: (old syntax):

```
vtk_resourcemap_set Priority:high UNLIMITED
vtk_resourcemap_set Queue:regression 2
vtk_resourcemap_set Queue:regression 2 "win10"
vtk_resourcemap_set Queue:regression 2 "win10 | linux"
vtk_resourcemap_set Queue:regression 2 "win10 | linux" 0 2h
```

vtk_resourcemap_set_in_bulk

Obsolete

Usage:
vtk_resourcemap_set_in_bulk

Description:
Obsolete procedure, that was meant to be used by MultiQueue. Superceded by
vov_multiqueue_set.

vtk_resourcemap_set_limit

Usage:
vtk_resourcemap_set_limit name value args

Description:
This procedure will set all limit resources that are derived from the one mentioned
as first argument, namely those where the @USER@ string has been replaced by a real
user name.

Options:
* -special <SPECIAL CASE> -- Specify exceptions to the limit
for different users. The SPECIAL_CASE argument
is a Tcl list consisting of an even number of elements
of the form "name number" (see example below).
The number cannot be "UNLIMITED".
It has to be a non-negative integer.

Example:
vtk_resourcemap_set_limit Limit:q_hsim_@USER@ 4
vtk_resourcemap_set_limit Limit:q_hsim_@USER@ 4 -special {
Limit:q_hsim_john 3
Limit:q_hsim_mary 5
}

vtk_resourcemap_set_limit_using_client

Usage:
vtk_resourcemap_set_limit <NAME> <VALUE> [OPTIONS]

Args:
name value args

Description:
This procedure will set all limit resources that are derived from the one mentioned
as first argument, namely those where the @USER@ string has been replaced by a real
user name.

Options:

```
-ooq    <LicenseResource>    -- A license resource we have to monitor
                                to detect out-of-queue uses, which
                                is used to enforce a combined in-queue and out-of-
queue
                                limit. This option can be repeated.
-special <SPECIAL_CASE>      -- Specify exceptions to the limit
                                for different users. The SPECIAL_CASE argument
                                is a Tcl list consisting of an even number of
elements
                                of the form "name number" (see example below).
                                The number cannot be "UNLIMITED".
                                It has to be a non-negative integer.
```

Example:

```
vtk_resourcemap_set_limit Limit:q_hsim_@USER@ 4
vtk_resourcemap_set_limit Limit:q_hsim_@USER@ 4 -special {
    Limit:q_hsim_john 3
    Limit:q_hsim_mary 5
}
vtk_resourcemap_set_limit Limit:q_hsim_@USER@ 4 -ooq License:hsim
```

vtk_resourcemap_set_revocation_delay

No information is available.

vtk_resourcemap_steal

Usage:

```
vtk_resourcemap_steal jobToStealResourcesId resourceName
```

Description:

This procedure is used in the context of job preemption. It is used to steal the specified resource from the job. Only ADMIN can execute this procedure. The resource is eliminated from the field GRABBEDRESOURCES and appended to the field STOLENRESOURCES.

Examples:

```
vtk_resourcemap_steal 0123456 License:sim#1
vtk_resourcemap_steal 0123456 License:sim#3
```

Returns:

Nothing.

See Also

[vtk_resourcemap_return](#)

[vtk_resourcemap_change_grab](#)

vtk_resources

vtk_resources_get

Obsolete

```
Usage:
vtk_resources_get

Description:
OBSOLETE: use vtk_generic_get resourcemaps instead.
Returns a formatted string with the summary of the available resources.

Returns:
A formatted text.
```

See Also

[vtk_generic_get](#)

See Also

[vtk_generic_get](#)

vtk_resources_parse

```
Usage:
vtk_resources_parse resource_expression_string

Description:
Parse the expression and print it in formatted mode or return an error.
Used to test resource expressions for correctness.

Returns:
An error code if it fails to parse the expression.
```

vtk_retrace

vtk_retrace

```
Usage          vtk_retrace cmd args

Description    Interface to vtk_retrace_check

get  -- Get all retraces

info id -display
info id -duration
```



```
info id -message  
info id -mode  
info id -name  
info id -percentdone  
info id -user
```

vtk_retrace_check

Usage:
vtk_retrace_check

Description:
Retrieves information about all existing retrace requests.

Returns:
A big formatted string.

vtk_retrace_format

Usage:
vtk_retrace_format retraceId

Description:

Example:
vtk_retrace_format 123445

```
-----  
| Retrace       : tmp:retrace:dir ${TOP}/src/doc  
| Id            : 04568415  
| Requested by: john@bison:0.0  
| Priority      : normal  
| Mode         : SAFE/CROSSBARRIERS  
| Work to do   : 14  tools  
| Status       : Is late.  
-----
```

Returns:
A formatted string

vtk_retrace_get

Usage:
vtk_retrace_get retraceId <array>

Description:

Get information about the given retrace.

Examples:

```
vtk_retrace_get 45678
1049577954 25s 988 14 {tmp:retrace:dir${TOP}/src/doc} john bison:0.0 {}
vtk_retrace_get 45678 array
1049577954 25s 988 14 {tmp:retrace:dir${TOP}/src/doc} john bison:0.0 {}
parray array
```

Returns:

A list of:

1. starttime
2. estimated duration
3. percent done
4. jobs to do
5. description
6. user
7. display
8. message

vtk_retrace_start

Usage:

```
vtk_retrace_start priorities mode target [aux]
```

Description:

This procedure supports the following arguments

1. priorities mode ALL
2. priorities mode NODE nodeId
3. priorities mode SET setId
4. priorities mode DIRECTORY dirName
5. (obsolete) priorities mode BACKWARD nodeId
6. (obsolete) priorities mode FORWARD nodeId
7. [-priorities <sched_priority[.exec_priority]>] [-mode <mode>] [-resources resourceList] [-target <target> <aux>] [-client] [-array <arrayname>] [-first] [-nop]

The argument priorities specifies both the scheduling and the execution priority using a "." to separate the two. You can specify only the scheduling priority. The default value for priority is "NORMAL.UNDEF".

You can also specify the priority with respect to the priority used last time for each job in the target, by using the specification "SAME", "INCR", or "DECR" (case insensitive).

The argument mode is a list of keywords from the following set: SAFE FAST AGGRESSIVE FORCE CROSSBARRIERS and at most one of these keywords used choose which files to check SKIPCHECK CHECKALL. Normally, only the primary inputs and the barriers are checked. With SKIPCHECK, no file is checked, with CHECKALL, all files in the upcone of the target is checked. The default value for mode is "SAFE".

Examples:

```
vtk_retrace_start NORMAL SAFE ALL
```

```
04568246 {} OK
vtk_retrace_start NORMAL.LOW SAFE ALL
04568246 {} OK
vtk_retrace_start HIGH.2 SET [vtk_set_find All:linux]
04568255 {} OK
vtk_retrace_start LOW "FORCE SKIPCHECK" BACKWARD $nodeId
04568259 {} ERROR
vtk_retrace_start -mode "FORCE SKIPCHECK" -target BACKWARD $nodeId
04568259 {} ERROR
```

Returns:

A list id msg status, where

- * id is the VovId of the retrace request
- * msg is a descriptive message
- * status is one of OK, CHANGE, NOTHING, ERROR.

For ease of use, however, we recommend using the `-array` option and refer to the fields returned in the array. See the following example.

```
retraceInfo(desc)           = a description
retraceInfo(display)        = bison:0.0
retraceInfo(id)             = 00092189
retraceInfo(message)        =
retraceInfo(mode)           = SAFE
retraceInfo(percent)        = 0
retraceInfo(priority,exec)   = 4
retraceInfo(priority,sched) = 4
retraceInfo(result)         = OK
retraceInfo(size)           = 1
retraceInfo(start)          = 1137786571
retraceInfo(status)         = IN PROGRESS
retraceInfo(user)           = someuser
retraceInfo(xdur)           = 10
```

vtk_retrace_start_job_or_set

Usage:

```
vtk_retrace_start_job_or_set [OPTIONS]
```

Description:

This is similar to `vtk_retrace_start`, but it is a bit more efficient and a bit less flexible. This procedure supports the following arguments

```
[-priority <sched_priority[.exec_priority]>] [-mode <mode>] [-resources
resourcelist] [-target <target> <aux>] [-first] [-nop]
```

You can also specify the priority with respect to the priority used last time for each job in the target, by using the specification "SAME", "INCR", or "DECR" (case insensitive).

The argument mode is a list of keywords from the following set: SAFE FAST AGGRESSIVE FORCE CROSSBARRIERS and at most one of these keywords used choose which files to check SKIPCHECK CHECKALL. Normally, only the primary inputs and the barriers are checked. With SKIPCHECK, no file is checked, with CHECKALL, all files in the upcone of the target is checked. The default value for mode is "SAFE".

Examples:

```
vtk_retrace_start_job_or_set -pri "NORMAL" -mode "SAFE" -targetid [vtk_set_find  
System:jobs]  
OK  
vtk_retrace_start_job_or_set -pri "NORMAL.LOW" -mode "SAFE" -targetid [vtk_set_find  
System:job]  
OK  
vtk_retrace_start_job_or_set -pri "HIGH.2" -targetid [vtk_set_find All:linux]  
OK  
vtk_retrace_start_job_or_set -pri "LOW" -mode "FORCE SKIPCHECK" -targetid $nodeId  
OK  
vtk_retrace_start -mode "FORCE SKIPCHECK" -targetid $nodeId
```

Returns:

A list OK or a descriptive error

vtk_retrace_stop

Usage:

```
vtk_retrace_stop retraceId | "-all" -why TEXT_REASON
```

Description:

Stop the given retrace. If the option -all is given instead of a retrace id, then all retrace requests are stopped, provided you have authorization to stop them. The optional reason is passed to eventually get set on the WHYSTATUS field and property of affected nodes.

Examples:

```
vtk_retrace_stop -all -why "Custom script stopping all retraces"  
vtk_retrace_stop 1234345  
vtk_retrace_stop 12345 45678 -why "Stop 2 retraces in one call"
```

Returns:

Nothing

vtk_sanity

vtk_sanity_check

Usage:

```
vtk_sanity_check setId_ignored REREAD|SANITY|RESOURCES|FAIRSHARE|TASKERS|NODESTATUS|  
PREEMPTION
```

Description:

Perform a sanity check of vovserver data structures. The first argument, setId, is ignored. The second argument, what, identifies the type of check to perform. If its value is 'SANITY' then vovserver performs a complete sanity check of all internal data structures. If its value is RESOURCES or FAIRSHARE or TASKERS or NODESTATUS or PREEMPTION, a check of the corresponding subsystem is performed. Otherwise (what = REREAD), vovserver rereads its configuration files, discards all caches, flushes log

and journal files, sanity checks resourcemaps, re-starts vovresourced, and checks VOV license validity.

Examples:

```
vtk_sanity_check 0 REREAD
vtk_sanity_check 0 SANITY
vtk_sanity_check 0 RESOURCES
vtk_sanity_check 0 PREEMPTION
```

vtk_select

vtk_select_create

Usage:

```
vtk_select_create [-select select_value] [-from from_value] [-where selectrule] [-order order_value] [-limit limit_value] [-cache 0|1] resultArray
```

Description:

Creates a data selection query of the vovserver using concepts similar to SQL queries. The results are organized as rows of columnar data. When any query created basic information about the query and number of rows, columns, etc. are returned via 'resultArray'. Generally, the results of a data results of a query are not returned via the 'resultArray' but can be later retrieved using vtk_select_get or iterated over with vtk_select_loop.

Options:

The query is specified by the following command line arguments:

```
* -select      -- comma separated list of fields to return
* -from        -- data source to which to apply the query to
* -where       -- selection criteria
* -order       -- comma separated list of fields by which to order the query
results
* -limit       -- maximum number of records to return
* -cache       -- flag to control cacheing of query.
```

To see a list of valid data sources, see the output of the following command:

```
vovselect objectname from objects
```

The -select values must be valid fields for the object type that corresponds to the -from sources. Additionally, any fields in the -where selection rule must only contain uses of fields that are also valid fields for the object type.

Example:

```
set select "USER,JOBNAME,CPUTIME"
set from "jobs"
set where "USER==bob"
set order "CPUTIME DESC"
set limit "100"
set cache "1"
```

```
set queryId [vtk_select_create -select $select -from $from -where $where -order $order -limit $limit -cache $cache queryResult]
```

```
puts "query id: $queryId"
query id: 000001037

parray queryResult
queryResult(bytes)      = 1428
queryResult(cache)      = 1
queryResult(colnames)   = USER JOBNAME CPUTIME
queryResult(errormsg)   =
queryResult(id)         = 000001037
queryResult(matched)    = 2
queryResult(numcols)    = 3
queryResult(numrows)    = 2
queryResult(ts,create)  = 1369054199
queryResult(ts,expire)  = 1369054209
queryResult(valid)      = 1
```

A full list of valid data sources can also be obtained by creating a query with -select value of objectname and a -from value of objects:

```
set queryId [vtk_select_create -select objectname -from objects queryResult]
```

Similarly, a list of fields for any specific object can be obtained by using a query:

```
set queryId [vtk_select_create -select fieldname -from objectname queryResult]
```

where objectname is replaced by the name of an actual supported object listed above.

Queries can either be cached or non-cached.

If it is known that the query will not generate very much data (e.g. a few megabytes of data at the most) the query can be specified to not be cached and the results returned during the query creation. This is generally a more efficient process since a persistent object is not created on the server. To specify a non-cached query, use the command-line argument '-cache 0'.

There is a command line utility that provides the same functionality as vtk_select based queries. The utility is called vovselect.

Returns:

The id of the query object. Also, the array queryResult, is populated containing basic information regarding the query.

vtk_select_delete

Usage:

```
vtk_select_delete queryId
```

Description:

This procedure is used to delete a query object from the server.

Example:

```
vtk_select_delete 00358286
```

Returns:

Success or an error message

vtk_select_get

Usage:

```
vtk_select_get queryId startRow endRow resultArray
```

Description:

Populates a TCL array with the values for rows [startRow:endRow] from the existing query given by the queryId. Note that the row indexes start at 1. A query that matched 100 rows would have it's first row numbered 1 and it's last row numbered 100. This is important for both specifying the range of rows to be returned as well as access values in the returned TCL array.

Example:

```
vtk_select_get 000001037 1 2 pqueryResult
```

```
parray pqueryResult
pqueryResult(1,CPUTIME)      = 100
pqueryResult(1,JOBNAME)     = VCS
pqueryResult(1,USER)        = bob
pqueryResult(2,CPUTIME)     = 60
pqueryResult(2,JOBNAME)    = spice
pqueryResult(2,USER)       = fred
pqueryResult(id)           = 000001037
pqueryResult(numviewrows)  = 2
pqueryResult(viewendrow)   = 2
pqueryResult(viewstartrow) = 1
```

Returns:

Success or an error message.

vtk_select_loop

Usage:

```
vtk_select_loop [-id query_id] [-select select_value] [-from from_value] [-where selectrule] [-order order_value] [-limit limit_value] [-cache 0|1] [-first firstRow] [-last lastRow] varlist script
```

Description:

vtk_select_loop provides a facility to loop over query results and execute a script block. A map of variables is provided that relates each column of data in a row to a variable in the script block. The variables are set to the appropriate column values before evaluating the script block. By default all rows are looped over. If it is desired to loop over a range of rows, then use the '-first' and '-last' options. The rows indexes start at 1. A query that matched 100 rows would have it's first row numbered 1 and it's last row numbered 100.

The query to be iterated over can be an existing query or a new query created by vtk_select_loop. To use an existing query, specify it via the '-id' option. To

create a new query, use the same options for `vtk_select_loop` that you would use with `vtk_select_create` to create the query.

See `vtk_select_create` for more information on the options to create a query.

Examples:

```
set select "USER, JOBNAME, CPUTIME"
set from "jobs"
set where "USER==bob"
set order "CPUTIME DESC"
set limit "100"
set cache "1"
```

```
set queryId [vtk_select_create -select $select -from $from -where $where -order
    $order -cache $cache queryResult]
```

```
parray queryResult
queryResult(bytes)      = 1428
queryResult(cache)      = 1
queryResult(colnames)   = USER JOBNAME CPUTIME
queryResult(errormsg)   =
queryResult(id)         = 000001037
queryResult(matched)    = 2
queryResult(numcols)    = 3
queryResult(numrows)    = 2
queryResult(ts,create)  = 1369054199
queryResult(ts,expire)  = 1369054209
queryResult(valid)      = 1
```

```
vtk_select_loop -id $queryId "userName jobName, runTime" {
    puts "User $userName had a $jobName job that ran for $runTime seconds."
}
```

User bob had a VCS job that ran for 100 seconds.
User fred had a spice job that ran for 60 seconds.

Returns:

Success or an error message.

vtk_selectrule

vtk_selectrule_validate

Usage:

```
vtk_selectrule_validate selectionRule
```

Description:

Check if a selection rule is valid.

vtk_server

vtk_server_config

Usage:

```
vtk_server_config variable value
```

Description:

Change selected variables in the server configuration, or perform administrative functions. Requires ADMIN privileges.

Examples:

```
vtk_server_config "autoRescheduleThreshold" 2
vtk_server_config "checkbarrier" -ignored-
vtk_server_config "httpSecure" 1
vtk_server_config "http.workerthreads" 5
vtk_server_config "http.proxytimeout" 3600
vtk_server_config "maxAgeRecentJobs" 2m
vtk_server_config "maxNormalClients" 400
vtk_server_config "maxNotifyClients" 100
vtk_server_config "netInfo" 1
vtk_server_config "resUserMatchTolerance" 2
vtk_server_config "taskerHeartbeat" 30
vtk_server_config "timeTolerance" 3
vtk_server_config "checklog" -ignored-
vtk_server_config "flushlogs" -ignored-
vtk_server_config "reopenlogs" -ignored-
vtk_server_config "suddenshutdown" PID_OF_VOVSERVER
```

See Also

[vtk_tasker_config](#)

[vtk_server_setenv](#)

vtk_server_dir

Usage:

```
vtk_server_dir flag
```

Description:

This procedure returns the server working directory for valid on the current host. To find the name of the server working directory on different hosts, user `vtk_swd_get`. The argument 'flag' is either "-physical" or "-logical". The value -expand is also supported for backwards compatibility and is equivalent to -physical.

Examples:

```
vtk_server_dir -physical
/home/john/projects/p1/ftadmin
vtk_server_dir -logical
$TOP/ftadmin
```

Returns
The name of the server working directory valid on the current host.

See Also

[7.97vtk_swd_get](#)

vtk_server_kill

Usage:
vtk_server_kill restartFlag

Description:

Example:
vtk_server_kill 0

Note:

ATTENTION! USE WITH CARE!

Send a message to the server to shutdown. The restartFlag is currently ignored. You have to be an ADMIN for this procedure to have an effect.

vtk_server_log

Usage:
vtk_server_log message

Description:
Write message to the server log.

Returns:
Nothing

vtk_server_setenv

Usage:
vtk_server_setenv variable value

Description:

Can be used by ADMIN to set an environment variable in the vovserver. Use at your own risk.

See Also

[vtk_server_unsetenv](#)

[vtk_server_config](#)

[vtk_generic_get](#)

vtk_server_unsetenv

Usage:
vtk_server_unsetenv variable

Description:
Can be used by an ADMIN to unset an environment variable in the vovserver. Use at your own risk. This is the same as vtk_server_setenv VARIABLE --delete--

See Also

[vtk_server_config](#)

[vtk_server_setenv](#)

[vtk_generic_get](#)

vtk_set

vtk_set_cache

Usage:
vtk_set_cache setId [OPTIONS]

Description:
Cache a set given its id.

Example:
vtk_set_cache 13221
vtk_set_cache 13221 -neighbors

vtk_set_cohort

Usage:

```
vtk_set_cohort setId boolean
```

Description:

If the boolean is true, all jobs in the set will be considered a "cohort", meaning that the scheduler will favor those jobs over the others.

Example:

```
vtk_set_cohort 12345 1  
vtk_set_cohort 12345 0
```

Returns:

Nothing

vtk_set_create

Usage:

```
vtk_set_create [OPTIONS] setName, rule
```

Description:

Options:

- * -transient ,to define the set as transient (no attach/detach events)
- * -client ,to define the set as client-set (will be forgotten when client is closed)
- * -universe SETID ,to define the universe set of the new set (default System:nodes)
- * -smart ,to have The set updated automatically based on selection rule.
- * -smartc ,to have a type of smart set computed only when a node is created.
- * -expire timespec ,For smart sets, this is the time after which the sets are discarded.
- * -noop ,Ignored option.
- * -hier ,to create all intermediate sets required for the true set hierarchy, based on the name of the set. The intermediate sets preserve the -transient and -client flags if present.

If the set name begins with the string "===" or with the string "tmp:", the set is considered transient.

Examples:

```
vtk_set_create All:linux "isjob resources~linux"  
293383  
vtk_set_create -transient tmp:upcone ""  
293390  
vtk_set_create -hier aa:bb:cc:dd ""  
221324  
vtk_set_create -client myset:upcone ""  
293397  
vtk_set_create -smart -expire 2m "Smart:nclist" "isjob user==abc dir~/mytest/"  
293399
```

Side Effect:

If an old set by the same name exists, it is destroyed.

Returns:
The setId of the new set or an error message.

vtk_set_find

Usage:
vtk_set_find setName

Description:
Find a set by name. Returns the setId or 0 if the set does not exist.

Examples:
vtk_set_find "System:nodes"
00000002
vtk_set_find "bad_set_name"
00000000

vtk_set_forget

Usage:
vtk_set_forget [-elements] [-hier] [-geo] [-max MAX_NODE_FORGOTTEN] setId

Description:
Forget the set. If option -elements is specified, the elements of the set will be forgotten instead. If option -hier is used, forget subsets also. If option -hier and -elements are used together, forget all elements in the set and subsets. If the option -max is used, then up to MAX elements are forgotten. The option only has effect in conjunction with the -elements option.

Examples:
vtk_set_forget -hier 29303
vtk_set_forget -hier -elements 29303
vtk_set_forget -elements 29303
vtk_set_forget -elements -max 1000 29303
vtk_set_forget -geo 29303
vtk_set_forget 29303

Returns:
'ok' if the set or its elements have been deleted, or 'error' if the set cannot be deleted because it is protected.

vtk_set_get

Usage:

```
vtk_set_get [-usecache] setId [resultArray]
```

Description:

Get information about a set of nodes. The procedure fails if it is passed an illegal setId. With just the setId this procedure returns a list consisting of the following elements:

1. setName
2. setSize
3. selectionRule
4. creationDateInt
5. creationDateChars

If the resultArray is provided, it is filled with similar information.

Examples:

```
vtk_set_get 24242
All:linux 3617 {isjob resources~linux} {Tue Mar 11 20:53:37 2003}
vtk_set_get 3 setInfo
parray setInfo
setInfo(creationdate)           = 1049429350
setInfo(flag:automatic)         = 0
setInfo(flag:client)            = 0
setInfo(flag:intable)           = 0
setInfo(flag:protected)         = 1
setInfo(flag:protectedelements) = 0
setInfo(flag:transient)         = 0
setInfo(isempty)                = 0
setInfo(name)                   = System:jobs
setInfo(owner)                  = john
setInfo(rule)                   =
setInfo(size)                   = 19777
```

vtk_set_get_elements

Usage:

```
vtk_set_get_elements setId formatString [OPTIONS]
```

Description:

Get a list of elements of the given set formatted according to the format string. The optional 'first' and 'last' arguments are in the range from 0, for the first element, to N-1 for the last elements in the set. Minus value will count from the end.

Options:

- * -normal, not to show sub jobs or system jobs.
- * -subjobs, to include sub jobs.
- * -systemjobs, to include system jobs .
- * -alljobs, to include both system jobs and sub jobs as well.
- * -hier, to include all elements in subsets.

```
* -subsets, to include subsets only, not elements.
* -rule, to include all elements that satisfy rule.
* -selrule, the same as -rule -first, to include all elements after the first index.
* -last, to include all elements before the last index.
```

Examples:

```
set setId [vtk_set_find System:jobs]
00000003
# Get all elements
vtk_set_get_elements $setId "@ID@ @STATUS@ @HOST@ @USER@ @COMMAND@"
{03691207 VALID bison john vov updateVersionHeader buildVersion.tcl}
{03691386 VALID noyce john vw cc -o pngtran.o -c pngtran.c}
...
# Get the first 20 elements
vtk_set_get_elements $setId "@ID@ @STATUS@ @HOST@ @USER@ @COMMAND@" -first 0 -last
19
# Get the last 20 elements
vtk_set_get_elements {$setId} @ID@ @STATUS@ @HOST@ @USER@ @COMMAND@ -20 -1
```

Returns:

A list of formatted element descriptions.

vtk_set_get_elements_chunks

Usage	vtk_set_get_elements_chunks
Description	DEPRECATED

vtk_set_get_elements_enhanced

Usage	vtk_set_get_elements_enhanced
Description	DEPRECATED. Please use vtk_set_get_elements.

vtk_set_get_or_create

Usage:
vtk_set_get_or_create setname selectionrule

Description:
If the set with setname exists, the procedure does vtk_set_find. If the set doesn't exist, it is the same as vtk_set_create.

vtk_set_operation

Usage:

```
vtk_set_operation setId nodeOrSetId operation
```

Description:

The first argument setId should be the ID of an existing set, which has been created by vtk_set_create.

If operation is CONTAINS, ATTACH or DETACH, the second argument represent a list of node id's.

If operation is UNION, INTERSECTION, COMPLEMENT, UPCONE or DOWNCONE the second argument must be the Id of a set. The COMPLEMENT operation is defined as the intersection of the second set with the complement of the first set.

If operation is INPUTS, OUTPUTS, the second argument is the reference set, the first argument is the accumulator.

If operation is CLEAR, the second argument is ignored.

The parameter operation is case-insensitive.

Returns:

1 if the operation returns OK, 0 otherwise. The return code is especially useful for the CONTAINS operation.

vtk_set_recompute

Usage:

```
vtk_set_recompute [-hier] setId
```

Description:

For sets with a selection rule, this procedure causes the server to recompute the set.

Example:

```
vtk_set_recompute 12345  
vtk_set_recompute -hier 12345
```

Returns:

Nothing

vtk_set_rename

Usage:

```
vtk_set_rename setId newSetName
```


Description:
Rename the given set.

The procedure fails if:

- * the set does not exist
- * the set is protected
- * the new name is reserved
- * there already exist a set by the same name.

vtk_set_report

Usage:
vtk_set_report [-hier] setId which array

Description:
Get a report by user, group, host, osgroup, jobclass or jobproj for a given set.

Example:
vtk_set_report 00012345 user reportInfo; parray reportInfo

vtk_set_set

Usage:
vtk_set_set setId array

Description:
array indices include name rule flag:transient flag:client flag:smart flag:smartc
which are Set Name, Rule, Transient and Client flags for the given set.

The procedure fails if:

- * the set does not exist
- * the set is protected
- * the new name is reserved
- * there already exist a set by the same name.

vtk_set_statistics

Usage:
vtk_set_statistics [-hier] [-levels] setId [array]

Description:
With just the setId argument, this procedure returns a list consisting of the following information:

1. setStatus (see info on set status)
2. number of nodes

3. number of files
4. number of jobs
5. total duration for all jobs in the set
6. number of jobs with unknown duration
7. {file_status_desc}
- 8 {job_status_desc}

The two lists file_status_desc and job_status_desc consists of an even number of elements, in which the first element represent a node status and the second element is the number of nodes with that status.

Examples:

```
vtk_set_statistics 3
FAILED 19777 0 19777 127510 1 {} {VALID 17170 INVALID 2595 FAILED 12 }
vtk_set_statistics 3 stats
parray stats
stats(duration)           = 127510
stats(files)              = 0
stats(files,BOGUS2)       = 0
stats(files,DELETED)      = 0
stats(files,EMPTY)        = 0
stats(files,FAILED)       = 0
stats(files,INVALID)      = 0
stats(files,MISSING)      = 0
stats(files,RETRACING)    = 0
stats(files,RUNNING)      = 0
stats(files,SLEEPING)     = 0
stats(files,UNKNOWN)      = 0
stats(files,VALID)        = 0
stats(jobs)               = 19777
stats(jobs,BOGUS2)        = 0
stats(jobs,DELETED)       = 0
stats(jobs,EMPTY)         = 0
stats(jobs,FAILED)        = 12
stats(jobs,INVALID)       = 2595
stats(jobs,INVALID,scheduled) = 17
stats(jobs,MISSING)       = 0
stats(jobs,RETRACING)     = 0
stats(jobs,RETRACING,suspended) = 0
stats(jobs,RUNNING)       = 0
stats(jobs,RUNNING,suspended) = 0
stats(jobs,SLEEPING)      = 0
stats(jobs,UNKNOWN)       = 0
stats(jobs,VALID)         = 17170
stats(nodes)              = 19777
stats(status)             = FAILED
stats(unknown)            = 1
stats(lastchange)         = 1076690458
```

vtk_set_subselect

Usage:
vtk_set_subselect [OPTIONS] setNameOrId selectionRule

Description:

Select a subset of the given set, according to the given selection rule. The original set can be specified either by name or by id.

Options:

- * -name name_of_new_set, to specify the name of the new set, otherwise a default name is chosen.
- * -setname name_of_new_set, same as -name.
- * -transient, to label the new set as transient, in order to disable the generation of uninteresting events.
- * -client, to label the new set as private to the calling client. The set will be automatically deleted when the client exits.
- * -hier, to include all elements in subsets.

Returns:

The id of the new subset.

vtk_setbrowser

vtk_setbrowser

Usage:

```
vtk_setbrowser Set browser widget
```

Description:

This procedure manages the set browser in the VOV console. It is designed to be used by the GUI developer, not by the regular user.

Example:

```
vtk_setbrowser <widget>  
vtk_setbrowser <widget> update
```

vtk_sign

vtk_sign_file

Usage:

```
vtk_sign_file string
```

Description:

Calculates CRC of file specified by argument.

vtk_signal

vtk_signal_handle

Usage:

```
vtk_signal_handle signal_name action
```

Description:

Specify handling of signals in a Tcl interpreter. The following signals can be handled: HUP INT QUIT CHLD TERM USR1 USR2 (case insensitive). The actions that can be performed with these signals are IGNORE DEFAULT CATCH (case insensitive). If the action is CATCH, when the signal is delivered to the process, the Tcl procedure VovSignalCatcher {\$sigNumber} {\$sigName} is called.

Examples:

```
vtk_signal_handle HUP IGNORE
vtk_signal_handle 1 IGNORE
vtk_signal_handle CHLD DEFAULT
vtk_signal_handle QUIT CATCH .
```

Returns:

Nothing

vtk_strings

vtk_strings_release

Usage:

```
vtk_strings_release
```

Description:

Release memory used for temporary strings. This procedure is used in daemons that communicate a lot with the server. Such communication allocates temporary strings which then need to be eliminated.

Example:

```
vtk_strings_release
```

vtk_substitute

Usage:

```
vtk_substitute text arrayname
```

Description:

Substitute the substrings in 'text' using an associative array as guide. For each name defined in array, replace the string name with the string \$array(\$name). The array is passed by reference to its name.

vtk_swd

vtk_swd_get

Usage:

```
vtk_swd_get key
```

Description:

This procedure helps locate the server working directory. If key is the null string, this returns the list of valid keys.

Examples:

```
vtk_swd_get ""  
unix registry canonical windows  
vtk_swd_get unix  
/export/proj/chip/ftadmin  
vtk_swd_get "windows"  
p:/chip/ftadmin
```

See Also

[vtk_server_dir](#)

vtk_swd_set

Usage

```
vtk_swd_set key path
```

Description

This procedure is used exclusively in the policy.tcl file.
It is used to define the various names for the server working directory.

Examples:

```
vtk_swd_set unix      /u/local/vnc  
vtk_swd_set windows h:/vnc
```

SEE ALSO: [vtk_server_dir](#) [vtk_swd_get](#)

vtk_tasker

vtk_tasker_attach_to_agent

Usage

```
vtk_tasker_attach_to_agent taskerId agentId
```

Description

This procedure is used to attach a BPS tasker to a BPS agent

Example

```
vtk_tasker_attach_to_agent 00358286 00358287 ok
```

Returns

"ok" or an error message

vtk_tasker_config

Usage vtk_tasker_config taskerIdOrName variable value

Description

WHERE: variable can be one of:

allowcoredump BOOLEAN, mostly used for debugging serious crashes (not gonna happen, anyways)

cpus INTEGER, sets the number of cpus(cores)

capacity INTEGER, sets the number of slots (but not more than the max capacity of the tasker)

changenamereconnect BOOLEAN, if true then the tasker will add an _r to its name in the case of reconnection to its vovserver; the default is false

cleanupchildprocesses BOOLEAN, if true kill all child processes when a job exits, requires taskers to support cgroups; default is false (Linux only)

close STRING, closes the tasker from accepting jobs, displaying the accompanying STRING until opened

coeff REAL in (0.01,100.0), that is the coefficient for the tasker (it is used to divide the raw power to compute the effective power)

debugjobcontrol BOOLEAN, used for debugging the job control activities (output in the tasker logs)

debugnuma BOOLEAN, used for debugging the NUMA affinity of jobs (output in the tasker log)

efftotram INTEGER, expresses the effective total RAM of a tasker, in MB

expirelogs BOOLEAN, simulate the expiration of the tasker logs, as it would normally happen at midnight; new logs are immediately created

rotatelog BOOLEAN, recreate a log file and directory only when missing.

liverecorder COMMAND in (on off save), for activating LiveRecorder by UndoDB

liverecorder.logsize INTEGER, to specify the max value of the log size in bytes

maxload REAL, the maximum load acceptable for the tasker (if the effective load passes this threshold, the tasker stop accepting jobs)

maxwaitnostart TIMESPEC, the maximum wait we allow for jobs that have trouble starting (meaning the fork()/exec() takes an enormous time); the default is 60 seconds

maxwaittoreconnect TIMESPEC, the maximum wait after losing connection to server before attempting reconnection again

message STRING, change the message for the tasker

message,sys STRING, change the system message for the tasker (same as 'message')

message,usr STRING, change the user message for the tasker

```
mindisk INTEGER, specify the minimum amount of disk space in MB or in percentage
(0%-99%, for example, 10%) in /tmp for the tasker to be active
minramfree INTEGER, specify the minimum amount in MB of free RAM for the tasker to be
active
name STRING, change the name of the tasker; illegal or duplicate names are silently
discarded
numabindtosocket BOOLEAN, if true (default) the NUMA code allows the process to roam
all cores in the same physical socket, else it binds the process to very specific
cores
open STRING, opens a closed tasker so that it will accept jobs, displaying the
accompanying STRING until overwritten by another message
printstatus IGNORED, causes the tasker to print its own status information to the log
file
ramsentry BOOLEAN, used to activate the RAM sentry functionality, i.e. the
functionality that suspends small jobs to allow large jobs to finish
rawpower INTEGER, change the raw power of the tasker
refresh IGNORED, discard all caches in tasker, including environments and user
credentials
resources STRING, set the resources for this tasker
retrychdir INTEGER, sets the number of times chdir() is to be tried (assumes the file
system is so stressed that even chdir() has trouble)
retrychdirbackoff REAL, for retry chdir() the interval between tries increases
according to this parameter
retrychdirsleep INTEGER, sleep time between the first and second attempt at chdir()
setenv VAR=VALUE, define or modify an environment variable in the top-level vovtasker
shutdowncancel IGNORED, used to rescue taskers that have been stopped (with STOP) but
have not yet exited because they are still running jobs
unsetenv VAR, unset the named environment variable
update INTEGER, set the update interval for the tasker, typically 60 seconds
verbose INTEGER in (0,4), set the verbosity level of the tasker
waitafterjrc INTEGER, seconds to wait after doing a job control action, except for EXT
(before doing another one, the tasker never stops)
waitafterjrcext INTEGER, seconds to wait after doing a EXT job control

and value can take different formats depending on the variable.
```

Examples:

```
vtk_tasker_config 00123456 maxload 3.3
vtk_tasker_config 00123456 capacity 1
```

vtk_tasker_create

Usage

```
vtk_tasker_create array
```

Description

The array contains the information of the tasker create, including:

```
host
name
group
taskertype (one of "normal", "BPS_tasker", "BPS_agent")
capacity
numjobs
```

```
power
timeleft
resources
status
curload
effload
maxload
message,usr
message,sys
```

The following are for BPS tasker only:

```
status
    0: OK-1: Warning-2: Unknown

time left
number of running jobs
power
```

Example:

```
set tasker(host)      alpaca
set tasker(capacity)  2
set tasker(resources) "finfarm RAM/100"
vtk_tasker_create tasker
00358286
```

Returns

The tasker Id of the tasker just created, or an error message.

vtk_tasker_define

Usage vtk_tasker_define

Description Used in taskers.tcl file. Please refer to the documentation.

vtk_tasker_delete

Usage
vtk_tasker_delete taskId

Description
This procedure is used to delete a tasker object from server. Other names for this procedure are vtk_tasker_del and vtk_tasker_forget.

Example:

```
vtk_tasker_delete 00358286
ok
```


Returns
"ok" or an error message

vtk_tasker_find

Usage
vtk_tasker_find name

Description

Example
vtk_tasker_find alpaca
00358285
vtk_tasker_find tiger
0

Returns
The tasker Id of the tasker with the specified name, or 0 not found.

vtk_tasker_get

Usage
vtk_tasker_get taskId array

Description
This procedure fills the array with the tasker properties.

Example:
vtk_tasker_get 00358294 tasker
ok
parray tasker
tasker(capacity) = 1
tasker(cpus) = 1
tasker(curload) = 0.16
tasker(effload) = 0.16
tasker(host) = alpaca
tasker(lastupdate) = 1159547044
tasker(maxload) = 1.50
tasker(message) =
tasker(name) = alpaca
tasker(numjobs) = 0
tasker(power) = 143678
tasker(resources) = alpaca:0.0 RAM/753 linux i686 unix
tasker(resources,spec) = alpaca:0.0 @RAM@ @VOVARCH@
tasker(taskertype) = normal
tasker(status) = READY
tasker(timeleft) = Unlim.
tasker(persistent) = 0

Returns
"ok" or an error message

vtk_tasker_get_colormap

No information is available for this procedure.

vtk_tasker_get_jobs

Usage
vtk_tasker_get_jobs taskerid

Description
Find the jobs currently running on a tasker.

Returns
A list of ids

vtk_tasker_modify

Usage
vtk_tasker_modify taskerId array

Description
This procedure modifies the tasker identified by taskerId with the properties set in the array. The usage of the array is the same as in vtk_tasker_create.

Example:
set tasker(host) alpaca
set tasker(capacity) 2
set tasker(resources) "finfarm RAM/100"
vtk_tasker_modify 00382732 tasker
ok

Returns
"ok" or an error message

vtk_tasker_nc

Usage

```
vtk_tasker_nc ncserver OPTIONS
```

Description

Define an indirect tasker used to send jobs to an Accelerator queue. This procedure is useful only in a taskers.tcl file. The parameter "ncserver" can be the string "automatic" to automatically discover the default Accelerator, or it could be an instance of Accelerator specified in the format of "nc_name@host[:port]", where "port" is optional but recommended.

Option:

-single-user

If 1, use the single-user script to interface to NC

Default: 0, meaning that the binary multi-user interface is used.

-name agent_name

The name of this indirect tasker (only letters, numbers, _, and - are allowed)

Default: NC

-resources res_list

The resources to provide to the jobs routed through this indirect tasker

Default: The resource 'NC'.

-autokillmethod direct|ncstop|vovstop

Specify autokill method for this tasker.

Default: direct

-capacity capacity

The number of slots in this tasker. This is the maximum number of jobs transferred to the NC queue at the same time. Typically in the order of a few hundreds, but you can use thousands.

Default: 200

-group group_name

The fairshare group that will be assigned to the jobs that are dispatched by this indirect tasker

Default: "-", which is to ignore this option, meaning no change to the group for jobs

-health health_specs

Define which health checks should be performed by the tasker before declaring itself ready. The health specs

consists of a string with the characters in "pdwuPDWU", where the lower case implies disabling and the upper case implies enabling a particular check.

Default: "pDwu", which enables only the disk space check.

-addjobres res_list

The resources that will be added to the jobs when dispatched by this indirect tasker

Default: "-", which is to ignore this option, meaning no addition to job resources

```
-resfilter res_list
    Filter on the tasker resources to determine whether to show a NC
    tasker as BPS tasker
    Default: "-", which is to ignore this option, meaning no filtering, show all
    taskers in NC

-showFarmTaskers
    If present, show farm vovtaskers, in addition to dispatcher
    vovtasker. Recommended only for small to medium-size farms.
    Default: Just show the dispatcher.

-nconfig path-to-file
    Use the given nconfig.tcl interface file to control the
    resources offered by the dispatcher vovtasker. Use only one
    of -resources and -nconfig. If both are given, -resources wins.
    Default: "-", which is to ignore this option
```

Examples:

```
vtk_tasker_nc automatic
vtk_tasker_nc vnc@comet:6271
vtk_tasker_nc automatic -showFarmTaskers
```

```
vtk_tasker_nc vnc@comet \;
  -resources "License:*" \
  -resfilter "linux License:hspice" \
  -addjobres "linux RAM/100" \
  -group production \
  -name Q_SPICE
```

vtk_tasker_reserve

Usage

```
vtk_tasker_reserve taskId [OPTIONS]
```

Description

Reserve a tasker. Without any option, this procedure clears the reservation of this tasker.

Options

```
-user    username    Reserve for specified user or users
-group   groupname    Reserve for specified fairshare group or groups
-osgroup groupname    Reserve for specified Unix (or OS) group or groups
-jobclass jobclass    Reserve for specified jobclass or jobclasses
-jobproj jobproj      Reserve for specified jobproj
-bucketid bucketid_list Reserve for specified bucket id or ids
-id      id_list      Reserve for specified id or ids
-duration timespec    Reserve this tasker for specified duration. forever is an
accepted value.
-start   timespec     Set reservation start time
```

```
-end    timespec    Set reservation end time
```

Examples:

```
vtk_tasker_reserve 230
ok
vtk_tasker_reserve 230 -user john -duration 3h
ok
vtk_tasker_reserve 230 -user john,mary -group alpha -start 1049827481 -duration 2w
```

Returns

"ok" or an error message

vtk_tasker_set_defaults

Usage

```
vtk_tasker_set_defaults
```

Description

Define default values for the options used by vtk_tasker_define.
Used in taskers.tcl file. Please refer to documentation.

vtk_tasker_set_timeleft

Usage

```
vtk_tasker_set_timeleft time_left
```

Description

This procedure is used only from within a vovtasker.
The argument specifies the amount of time left for the tasker,
so that the server can decide to send jobs whose expected duration
does not exceed the time left.
If the time_left is "UNLIMITED", there is no limit on the time.
If the time_left is 0, the tasker is effectively suspended.

Where time_left is a time specification or the string "UNLIMITED".

Returns

Nothing.

vtk_tasker_vnc

Obsolete

Usage

```
vtk_tasker_vnc resources name
```

Description

Obsolete procedure. Use `vtk_tasker_nc` instead.

vtk_taskerlist

No information for this procedure

vtk_taskerlists

No information for this procedure

vtk_taskerset

vtk_taskerset_get

```
Usage:
vtk_taskerset_get [OPTIONS] [array]

Description:

Options:
-hwreq HW_REQUEST -- Show only taskers that satisfy the given HW request.
-rule SELRULE -- Show only taskers that match the given selection rule.
With no parameters, this procedure returns the list of all tasker ids.
With one parameter, such parameter represents the name of an associative array and
the procedure fills the array with information about all the taskers.

Examples:
vtk_taskerset_get
04544112 04544113 04544115 04544116 04544118 04544120 04544121
vtk_taskerset_get taskerInfo
04544112 04544113 04544115 04544116 04544118 04544120 04544121
parray taskerInfo
taskerInfo(04544112,capacity)      = 1
taskerInfo(04544112,cpus)         = 1
taskerInfo(04544112,curload)      = 0.33
taskerInfo(04544112,effload)      = 0.33
taskerInfo(04544112,host)         = mars
taskerInfo(04544112,lastupdate)   = 1159547044
taskerInfo(04544112,maxload)      = 1.50
taskerInfo(04544112,message)      =
taskerInfo(04544112,name)         = mars
taskerInfo(04544112,numjobs)      = 0
taskerInfo(04544112,power)        = 31080
taskerInfo(04544112,resources)    = mars RAM/128 sun5 sun4u unix
taskerInfo(04544112,taskertype)   = normal
taskerInfo(04544112,status)       = READY
taskerInfo(04544112,suspended)    = 0
taskerInfo(04544112,timeleft)     = Unlim.
```

...

vtk_tcl

vtk_tcl_global_eval

Usage:
vtk_tcl_global_eval script

Description:
Access to Tcl_GlobalEval()

vtk_time

vtk_time_pp

Usage:
vtk_time_pp time

Description:
Pretty-print a time expressed in seconds. This procedure is used a lot in CGI scripts. The inverse function is provided by the procedure VovParseTimeSpec.

Example:
vtk_time_pp 61
1m1s
vtk_time_pp 6.1e1
1m1s

vtk_time_psp

Usage:
vtk_time_psp time

Description:
Pretty-short-print a time expressed in seconds.

vtk_timestamp

vtk_timestamp_pp

Usage:
vtk_timestamp_pp time

Description

Example:
vtk_timestamp_pp [clock seconds]
20090505_091244

Returns
A pretty-printed timestamp.

vtk_trace

vtk_trace

Usage:
vtk_trace <close|open|status|switchto> ...

Description:
Procedure to control connection to the trace server.

Examples:
vtk_trace status
connected
vtk_trace close
vtk_trace open \$env (VOV_PROJECT_NAME) \$env (VOV_HOST_NAME)
vtk_trace switchto local
vtk_trace switchto client
vtk_trace status
disconnected
vtk_trace open \$projectname \$host [LOCAL | CLIENT]

vtk_trace_checkall

Usage:
vtk_trace_checkall primaryInputsOnlyFlag

Description:
Check all places. If the flag is 1, only the primary inputs are checked.

Returns:
Nothing

vtk_trace_dump_nodeset

Usage:
vtk_trace_dump_nodeset setName/setId fileName [tcl|make|csh|ps]

Description:

vtk_trace_is_connected

Usage:
vtk_trace_is_connected

Description:

Returns:
1 if the trace is connected to a server, 0 otherwise.

vtk_trace_list_places

Usage:
vtk_trace_list_places regularExpression

Description:

Returns:
A list of places in the default format.

Note:
This is an obsolete procedure. Use vtk_set_create instead to find places that match a given regular expression.

vtk_trace_list_sets

```
Usage:
vtk_trace_list_sets [OPTIONS]

Description:
List all sets in the trace.
The following options are mutually exclusive
* -ids      -- Show only the id's
* -names    -- Show only names (default)
* -all      -- List pairs of (id,name)
* -rx RX    -- Limit list to sets that match the specified regular expression RX
* -max N    -- Limit number of items returned. Useful in cases where there is an
               overflow caused by too many sets

Examples:
vtk_trace_list_sets -ids
000000002 000000003 000000004 01615042 01615043 01615049
vtk_trace_list_sets -all
{000000002 {System:nodes}} {000000003 {System:jobs}} ...
vtk_trace_list_sets -all -rx System:
{000000002 {System:nodes}} {000000003 {System:jobs}} ...
vtk_trace_list_sets -max 1000 -rx abc
...
```

vtk_trace_list_users

```
Usage:
vtk_trace_list_users

Description:
List all users in the trace.

Returns:
A list of id of users.
```

vtk_trace_rename

```
Usage:
vtk_trace_rename setName oldString newString

Description:
Replace all occurrences of oldString to newString for all nodes in the set with name
setName.

Example:
```

```
vtk_trace_rename System:jobs toolx tooly
Renaming means doing a string substitution
in the names of files and in the command lines
and cwd of tools:
Set Name      ""
Old String    "toolx"
New String    "tooly"
19777 nodes have been examined.
0 files have been modified.
2 tools have been modified.
```

Returns:
A long and somewhat useless preformatted string.

vtk_trace_reopen

Usage:
vtk_trace_reopen timeout

Description:
This procedure is used to reconnect to a server after a crash.

vtk_trace_save

Usage:
vtk_trace_save -dir savedir -nofork -time maxAllowedTime

Description:
Send a request to the server to save the trace database to disk. The actual saving may take an unspecified amount of time depending on the size of the trace. If option -nofork is used, this procedure returns after the trace has been save, else the procedure returns immediately, meaning that upon return the trace has not yet been saved.

Examples:
vtk_trace_save
vtk_trace_save -nofork
vtk_trace_save -nofork -time 5m
vtk_trace_save -time 5m -dir /tmp/foobar/savedir

vtk_transition

vtk_transition_add

Usage:

```
vtk_transition_add dir envName resources cmd inputList outputList
```

Description:

This procedure is controlled by `make(ctrl,action)`, which can be one of 'trace', 'script', or 'echo'.

If `make(ctrl,action)` is 'trace', this procedure is simply an interface to `vtk_transition_add_to_trace`, which takes the same arguments. This procedure tries to add the specified transition to the trace. If an identical transition already exists, its resources will be augmented, if necessary. At least all the files specified in ``inNameList'` and ``outNameList'` will have arcs to/from this transition.

If `make(ctrl,action)` is 'script', the procedure prints on stdout an executable script valid for the architecture `make(arch)`.

If `make(ctrl,action)` is 'echo', the transition is echoed on stdout.

The expected duration of the new transition is set to `make(xdur)`.

The autoforget flag is controlled with `make(autoforget)`.

The autokill is not a flag anymore but a time interval, and is controlled with `make(autokill)`.

The preemptable flag is controlled with `make(preemptable)`.

The jobclass is `make(jobclass)`.

The jobname is `make(jobname)`.

The jobkey is `make(jobkey)`.

Old transitions with output arcs to files in ``outNameList'` are deleted.

Side Effects:

`make(transId)` is set.

Returns:

`make(transId)` or 0 if the transition could not be created.

vtk_transition_add_to_trace

Usage:

```
vtk_transition_add_to_trace [OPTIONS] directory environment resources command  
inputFiles outputFiles
```

Description:

Option:

* -xdur time	-- Set the expected duration of the job.
* -autoflow bool	-- Set the autoflow flag.
* -autoforget bool	-- Set the autoforget flag.
* -autokill time	-- Set the autokill time
* -jobclass name	-- Set the job class
* -jobname name	-- Set the name of the job
* -jobproj name	-- Set the project of the job, used for accounting purposes
* -alreadycanonical bool	-- Assume that the directory is already canonical

```
* -preemptable bool      -- The job is preemptable
* -migratable bool      -- The job is migratable
* -multiphase bool      -- The job is multiphase
* -sticky bool          -- The I/Os declared are sticky
* -nonexec bool         -- The job is non-executable
* -jpp jpp              -- Specify a job placement policy
* -jk jk                -- Generate a security key valid for the duration of the
job
* -runmode runmode      -- Specify the run mode: normal, xterm_open, xterm_icon,
xterm_none
* -uniqueid bool        -- Force job to have a unique id by changing the
environment, if required
* -usesid bool          -- Use sessionid of processes to determine the processes
that belong to a job
* -fstokens tokens      -- Set fstokens for the job (default 1)
* -scheddate UnixTimestamp -- Do not start tool before given timestamp (default 0)
* -start integer        -- Job array start index
* -finish integer       -- Job array finish index
* -incr integer         -- Job array increment
* -snapshot string      -- Job environment snapshot property
* -origargs bool        -- Save job original arguments
* -logfile string       -- Name of the place
```

The -logoptions can be:

```
* -shared, if the output is shared among many jobs; all jobs must declare the output
as shared
* -force, if output conflicts are to be ignored (this is the default)
* -normal, resets all previous options
```

All nodes referenced in this procedure are added to the set with name make(setname). Input conflicts are ignored, output conflicts are forced, cycle conflicts are serious errors.

Create a job array. Place all created jobs in new set 'JobArray:setId'. For each job, the index is incremented from 'start' to 'finish' by 'increment'. The procedure fails if the total number of jobs is greater than the allowed limit.

Returns:

The id of the transition that has been added.
The id and name of the set containing the array jobs

Note:

This procedure is normally accessed by means of vtk_transition_add which is a Tcl wrapper.
And nc run tcl proc submitArrayJobs

vtk_transition_array

Usage:

```
vtk_transition_array setid refjobid start finish increment
```

Description:

Create a job array. Place all created jobs in the given set. For each job, the index is incremented from 'start' to 'finish' by 'increment'. The procedure fails if the total number of jobs is greater than the allowed limit.

vtk_transition_chown_to_me

Usage:
vtk_transition_chown_to_me jobId

Description:
Change owner (user) of the given job to the caller.

Example:
vtk_transition_chown_to_me 00123456

vtk_transition_color_control

Usage:
vtk_transition_color_control ACTION, ...

Description:
Control the color used to display VALID/FAILED jobs depending on the exit status.
ACTION may be one of RESET_VALID, GET_VALID, SET_VALID, RESET_FAILED, GET_FAILED, SET_FAILED Used only in the project.swd/gui.tcl configuration file. To be effective, it must be present in gui.tcl before vovconsole starts.

Examples:
vtk_transition_color_control RESET_VALID
vtk_transition_color_control GET_VALID a; parray a
vtk_transition_color_control SET_VALID 10 pink
vtk_transition_color_control RESET_FAILED
vtk_transition_color_control GET_FAILED a; parray a
vtk_transition_color_control SET_FAILED 1 blue

vtk_transition_create_resumerjob

Usage:
vtk_transition_create_resumerjob jobToResumeId

Description:
Create the resumer job for a suspended one. For internal use only. Used by vovpreemptd.

Returns:
resumer job Id.

vtk_transition_dispatch

Usage:

```
vtk_transition_dispatch jobId taskerId
```

Description:

Manually force a dispatch of a job to the given tasker. If option -delayed is used, then only the tasker is chosen as the tasker for the job, but the job will be dispatched only when all resource constraints have been satisfied. RETURN: Ok or {Illegal Id} or {Non exec job} or Unauthorized

vtk_transition_failcode_pp

Usage:

```
vtk_transition_failcode_pp mask
```

Description:

Given a bit mask representing the failure modes for a job, this procedure returns a tab separated list of descriptive strings.

Example:

```
vtk_transition_failcode_pp 0x101  
"Preempted by owner\tBad exit status\t"
```

vtk_transition_find

Usage:

```
vtk_transition_find environment directory command
```

Description:

Returns:

The ID of the transition, or 0 if not found.

vtk_transition_fire

Usage:

```
vtk_transition_fire jobId
```

Description:

Fire a transition given its job id. Then wait for it to complete.

The global variable `fireCode` is modified and can be used in case of error. It is assumed that we are already in the correct environment. This procedure uses the same code used by `vovtasker` to fire the transition. It is recommended that you use this procedure instead of trying to escape the command line in a way suitable for use by either `'exec'` or `/bin/sh`.

Note:

We now use `fireCode` instead of `errorCode`.

We do not use `errorCode` because that is a Tcl variable and we do not seem to have control of it.

It is assumed that we are already in the correct environment.

This procedure uses the same code used by `vovtasker` to fire the transition.

It is recommended that you use this procedure instead of trying to escape the command line in a way suitable for use by either `'exec'` or `/bin/sh`.

vtk_transition_get

Usage:

```
vtk_transition_get transitionId [array]
```

Description:

Returns information about a transition (i.e. a job). If the `transitionId` doesn't exist, an error is returned. If the `transitionId` refers to a job that has been retraced, the new id for the job is automatically used.

The command has two forms, with and without an array specification. The form without the array specification is deprecated because it provides much less information and in a list that is difficult to parse. We recommend you use only the form with the array argument.

Deprecated form: If array is not specified, a list of the following elements is returned:

1. node status
2. BARRIER or NORMAL (always normal)
3. environment
4. working directory
5. command line
6. resources
7. legal exits
8. user
9. host
10. exit status
11. start date
12. end date
13. duration
14. expected duration

Preferred form: if array is specified, it is filled with information about the transition. Below is an example of usage of this form.

```
vtk_transition_get 12345 jobInfo
parray jobInfo
jobInfo(autoflow)           = 0
jobInfo(autoforget)         = 0
jobInfo(autokill)           = 0
```



```
jobInfo(barrier)           = NORMAL
jobInfo(barrierinvalid)    = 0
jobInfo(cmd)               = clevercopy -m 644 FTaux.pdf ../doc/html/pdf
jobInfo(cputime)           = 0
jobInfo(dir)               = ${BUILD_TOP}/src/doc/html
jobInfo(duration)          = 1
jobInfo(end)               = 1049580600
jobInfo(env)               = BASE+SUPPORT
jobInfo(exit)              = 0
jobInfo(group)             = users
jobInfo(host)              = alpaca
jobInfo(legalexit)         = 0
jobInfo(maxram)            = 0
jobInfo(nodets)            = 1049580600
jobInfo(priority)          = 0
jobInfo(qtime)             = 0
jobInfo(res)               = htmldoc linux
jobInfo(retracingid)       = 00000000
jobInfo(scheduled)         = 0
jobInfo(taskerid)          = 00000000
jobInfo(start)             = 1049580599
jobInfo(status)            = VALID
jobInfo(tool)              = clevercopy
jobInfo(unsafe)            = 0
jobInfo(user)              = casotto
jobInfo(userxdur)          = -1
jobInfo(xdur)              = 1
```

vtk_transition_get_failcode_mask

No information available for this procedure.

vtk_transition_get_input_names

No information available for this procedure.

vtk_transition_get_io_names

```
Usage:
vtk_transition_get_io_names jobid mask regexp

Description:
For the given transition, give the list of input and outputs that match the given
regular expression.

Examples:
vtk_transition_get_io_names 12345 I {csh$}
vtk_transition_get_io_names 12345 IO chip
```

```
vtk_transition_get_io_names 12345 0 std
```

vtk_transition_get_or_create

Usage:

```
vtk_transition_get_or_create env cwd command resources
```

Description:

Also called `vtk_transition_find_or_create`, where

- * `env` is the environment
- * `cwd` is the working directory (need not be canonical)
- * `command` is the complete command line
- * `resource` is the resource list

vtk_transition_get_output_names

No information available for this procedure.

vtk_transition_operation

Usage:

```
vtk_transition_operation jobId operation_method_or_plan
```

Description:

Operation the job with the specified method.

Example:

```
vtk_transition_operation ${jobId} saveProfile
```

vtk_transition_preempt

Usage:

```
vtk_transition_preempt jobId [-noop] [-method METHOD_OR_PLAN] [-manualresume] [-resumeres RESLIST]
```

Description:

Preempt the job with the specified method.

Example:

```
vtk_transition_preempt $jobId -method SUSPEND -resumeres "License:abc" -manualresume
```

```
vtk_transition_preempt $jobId -method "BEGIN:RETRACING:EXT,USR1,vovsh  
10:SUSPENDED:LMREMOVE 20:LMREMOVED:DONE"
```

vtk_transition_set

Usage:

```
vtk_transition_set transitionId array
```

Description

Modify fields of a transition (job).

Uses an array of the same format used in `vtk_transition_get`.

Typically called after calling that function and modifying something in the array.

For example:

```
vtk_transition_get 12345 jobInfo  
set_jobInfo(legalexit) "0 1"  
set_jobInfo(res) "htmldoc linux"  
vtk_transition_set 12345 jobInfo
```

There are limitations on what can be changed and when.

Also, some changes have consequences including invalidating the job, marking the job unsafe, and moving the job to the correct fairshare bucket (taking the changes into account). Many fields can't be changed while a job is running.

The following fields can be changed any time, including when the job is running.

- * autokill (duration job runs before being killed automatically)
- * fstokens (fairshare {tokens;} when changed, moves the job to the proper fairshare bucket)
- * iscohort (can only be changed by job's owner)
- * isdistributed (can only be changed by job's owner)
- * jobname (can only be changed by job's owner)
- * jpp (job placement policy)
- * numa (job placement non-uniform memory access {policy;} linux only)
- * legalexit (if the job is not running, the exit status is checked again with the new legal {values;} if it does not match, the job is invalidated)
- * preemptable (flag indicating the job can be preempted)
- * priority (schedule priority)
- * runmode (for example, `xterm_iconic`)
- * schedfirst (schedule first flag)
- * statuslogging (logging of node status {changes;} NONE|ALL|INVALID|DOWNCONE)
- * subjob (flag that indicates it is a sub job; not normally modified this way)
- * systemjob (flag that indicates it is a system job; not normally modified this way)
- * xdur (expected duration)
- * xpriority (execution priority)

The following fields cannot be changed while the job is running. They can be changed by the job's owner or a project admin:

- * autoflow (flag that indicates a job should be {skipped;} if changed, job is moved to proper bucket)
- * end (end date/time of {job;} can only be changed for non-executable {jobs;} marks job unsafe)
- * ephemeral (flag that indicates the job is temporary and will go away after running)

```
* exit (exit status {code;} can only be changed for non-executable {jobs;} marks job
unsafe)
* host (execution host)
* nojournal (flag that turns off journal entries for the job)
* profile (flag that turns on profiling for the job)
* qtime (time job spent queued)
* res {(resources;} if changed for a scheduled job, the job is moved to the proper
bucket)
* scheddate (date/time job was scheduled)
* start (start date/time of {job;} can only be changed for non-executable {jobs;}
marks job unsafe)
```

The following fields cannot be changed while the job is running. They can only be changed by the job's owner (not a project admin):

```
* autoforget (flag that indicates the job will be forgotten after completion)
* cmd (command line of job; job is invalidated if changed)
* deadline (the desired date/time the job should be completed)
* dir (filesystem path where job runs; job is invalidated if changed)
* env (named environment of job; job is invalidated if changed)
* group (fairshare group; if changed for a scheduled job, the job is moved to the
proper bucket)
* jobclass (resource group)
* jobproj (project associated with job)
* osgroup (operating system group; can only be changed for non-executable jobs)
* res,aux (aux resources; if changed for a scheduled job, the job is moved to the
proper bucket)
* submithost (host from which job is submitted)
* tool (name of tool associated with command)
* user (operating system user; can only be changed for non-executable jobs)
```

The following field can only be changed while the job is RETRACING (not RUNNING):

```
* transfer (flag that indicates the job is in transfer to a tasker; causes smart
sets to be recomputed)
```

Returns:

0 if change was made or no changes were needed; 1 if there was an error; also one of the following messages: Job modified, Cannot modify, Not authorized, No change to be applied, Bad group name

vtk_transition_set_failcode

Usage:

```
vtk_transition_set_failcode jobId mask_or_long_char_code
```

Description:

Add the specified codes specified in 'mask' to the specified job.

Example:

```
vtk_transition_set_failcode $jobId 0x101
vtk_transition_set_failcode $jobId "no outputs"
```

vtk_transition_update_stats

Usage:

```
vtk_transition_update_stats transitionId stat1 value1 ... ..
```

Description:

WHERE stat? is one of MAXVSIZE MAXRSS CURRSS MAXPGFLT TOTCPU TIME IS_SUSPENDED PID value is a 32-bit integer quantity Also supported are these values for "stat", which are followed by 64-bit values: READCNT WRITECNT TOTCPU TIME64. TOTCPU TIME and TOTCPU TIME64 are expressed in milliseconds. MAXRSS CURRSS MAXVSIZE are in MB. READCNT WRITECNT are in bytes.

Also supported are these values for "stat", which are followed by 64-bit values: READCNT WRITECNT TOTCPU TIME64.

TOTCPU TIME and TOTCPU TIME64 are expressed in milliseconds. MAXRSS CURRSS MAXVSIZE are in MB. READCNT WRITECNT are in bytes.

vtk_umask

vtk_umask_get

Usage:

```
vtk_umask_get
```

Description:

Interface to System's umask(mask).

Returns:

Current value of umask.

vtk_umask_set

Usage:

```
vtk_umask_set
```

Description:

Interface to System's umask(mask).

Returns:

Previous value of umask.

vtk_unixgroup

vtk_unixgroup_get_users

vtk_unixgroup_get_users

Usage:

```
vtk_unixgroup_get_users
```

Description:

Get the users that belong to a give unix group. The group can be specified either by name or by id. An error is returned if the group does not exist.

vtk_user

vtk_user_find

Usage:

```
vtk_user_find userName [array]
```

Description:

Return the VovId of a user given a user name. If the optional argument array is given, the array is filled with information as in vtk_user_get.

Examples:

```
vtk_user_find john  
03072450  
vtk_user_find john a  
03072450  
parray a  
a(name) = john  
a(id) = 03072450  
...
```

Returns:

The ID of the user or 00000000 if the user does not exist.

See Also

[vtk_trace_list_users](#)

vtk_user_forget

Usage:
vtk_user_forget userId

Description:
Forget the user with given ID. Only the ADMIN can use this procedure.

Returns:
TCL_OK on success and TCL_ERROR on failure.

vtk_user_get

Usage:
vtk_user_get userId array

Description:
Returns information about a user. If the ID doesn't exist, an error is returned. The associative array is filled with information about the specified user, including:
user name
user default priority
user max priority
date of last connection to the server
number of executed jobs
number of services required
used time

Example:
set uid [lindex [vtk_trace_list_users] 0]
vtk_user_get \$uid userInfo
parray userInfo
userInfo(defaultpriority) = 0
userInfo(executedjobs) = 773898
userInfo(lastconnectiontime) = 1049502482
userInfo(lastjobtime) = 1049502242
userInfo(maxpriority) = 0
userInfo(name) = casotto
userInfo(services) = 0
userInfo(servicetime) = 1090811968

See Also

[vtk_trace_list_users](#)

vtk_user_get_shell

Usage:

```
vtk_user_get_shell
```

Description:

Get the default shell for the current user from the /etc/passwd entry. This is used, for example, in vovtaskermgr start to determine how to start the remote taskers based on the user's shell.

Example:

```
vtk_user_get_shell  
/bin/csh
```

See Also

[7.55vtk_logname](#)

[7.38vtk_groupname](#)

vtk_user_modify

No information for this procedure.

vtk_user_security

Usage:

```
vtk_user_security
```

Description:

Returns a pair {n s} representing the security code in integer and string format.

Example:

```
vtk_user_security  
5 ADMIN
```

vtk_user_set

Usage

```
vtk_user_set userName [OPTIONS]
```

Description

This procedure is used exclusively inside the policy.tcl file. The supported options are:

-defaultGroup	groupName	Used to specify the default group for the user
-group	groupName	Used to specify the group for which the fairshare weight applies.

assumed.	If the option is not used, the default group "users" is
-weight N	Used to specify the fairshare weight of the user in the
group specified by	the option -group.
-windowsName name	Used to specify an alternative name for the user to be
used on Windows.	

Examples

```
vtk_user_set john -defaultGroup regression -weight 10
vtk_user_set john -group alpha -weight 25
```

vtk_user_set_http_password

Usage:
vtk_user_set_http_password UNDOCUMENTED

Description:
This is a preliminary implementation. The procedure takes 3 arguments: ""
"winpassword" "ftp-password"
Also used to set the Windows Password as vtk_user_set_win_password.
To clear a password, pass the string "--clear--".

Examples:

Set and clear the FlowTracer password:
vtk_user_set_http_password "" "" "abc1234"
vtk_user_set_http_password "" "" "--clear--"

Set and clear the windows password.
vtk_user_set_http_password "" "xyz6789" ""
vtk_user_set_http_password "" "--clear--" ""

vtk_usergroup

vtk_usergroup_add

Usage:
vtk_usergroup_add userGroupName userList

Description:
Adds the specified users to the specified user group. If the user group with than
name does not exist, it is created.

Example:

Adds users 'andy' and 'barney' to the user group called 'designers'.

```
vtk_usergroup_add designers andy barney
```

vtk_usergroup_contains

Usage:

```
vtk_usergroup_contains userGroupName userName
```

Description:

Returns 1 or 0 if the user group contains the user.

Example:

```
vtk_usergroup_populate designers joe bob sally ralph  
set isContained [vtk_usergroup_contains designers joe]; # Returns 1.  
set isContained [vtk_usergroup_contains designers cathy]; # Returns 0.
```

vtk_usergroup_delete

Usage:

```
vtk_usergroup_delete userGroupName
```

Description:

Deletes the named user group.

Example:

```
vtk_usergroup_delete designers
```

vtk_usergroup_get

Usage:

```
vtk_usergroup_get userGroupName array
```

Description:

Returns information about the specified user group.

Example:

```
vtk_usergroup_populate designers joe bob sally ralph  
vtk_usergroup_get designers array  
parray array  
array(usergroup) = designers  
array(count)     = 4  
array(id)        = 000047193
```

```
array(0,user)    = joe  
array(1,user)    = bob  
array(2,user)    = sally  
array(3,user)    = ralph
```

vtk_usergroup_getgroups

Usage:
vtk_usergroup_getgroups array

Description:
Returns the list of all user groups.

Example:
vtk_usergroup_populate designers joe bob sally ralph
vtk_usergroup_populate coders brian swami
vtk_usergroup_getgroups array
parray array
array(count) = 2
array(0,usergroup) = designers
array(1,usergroup) = coders

vtk_usergroup_populate

Usage:
vtk_usergroup_populate userGroupName userList

Description:
Populates a new user group with the list of users provided. If a user group with that name already exists, the specified set of users overwrites any existing users in the user group.

Example:
vtk_usergroup_populate designers joe bob sally ralph

vtk_usergroup_remove

Usage:
vtk_usergroup_remove userGroupName userList

Description:
Removes specified users from user group.

Example:

Removes user 'ralph' from the user group named 'designers'.

```
vtk_usergroup_remove designers ralph
```

vtk_ftime

Usage:

```
vtk_ftime file, actime, modtime
```

Description:

Interface to System's utime(file, utimbuf).

vtk_validate_keyformat

Usage

```
vtk_validate_keyformat key
```

Description

Validate the format of a key to see if it appears to be valid.
i.e. it appears to be of the format: "32:ABCDEFGH" with a size
prefix followed by base64 encoded bytes, and that
the decoded byte size matches the declared size.

Example

```
vtk_validate_keyformat "32:WLnCP2UrB5AxeasboN03Zo8o6F4fu4CPknVINqen9Zg="
```

Seealso

```
vtk_generate_signature_keypair vtk_validate_message
```

Returns

1 indicating message validation was a success, or 0 indicating it was not

vtk_wave

vtk_wave_addinterval

Obsolete

Usage:
vtk_wave_addinterval waveGroupName waveIndex t1 t2 value

Description:
OBSOLETE: please use vtkwave_addinterval instead.

vtk_wave_addpoints

Obsolete

Usage:
vtk_wave_addpoints waveName Index TimeValueList

Description:
OBSOLETE: use vtkwave_addpoints instead.

vtk_wave_crop

Obsolete

Usage:
vtk_wave_crop waveName startTs endTs

Description:
Crop the wave between startTs and endTs. OBSOLETE: use vtkwave_crop instead.

vtk_wave_load

Obsolete

Usage:
vtk_wave_load waveGroupName waveFile [start [end]]

Description:
Load a wave from the given file, limiting to the interval from start to end (default now).

OBSOLETE: Use vtkwave_load instead.

vtk_wave_multiply

Obsolete

```
Usage:
vtk_wave_multiply waveGroup result factor1 factor2

Description:
resultRes: the name for to store the results

OBSOLETE: Use vtkwave_multiply or vtkwave_op instead.
```

vtk_wave_remove_glitches

Obsolete

```
Usage:
vtk_wave_remove_glitches waveGroupName waveIndex maxGlitchWidth

Description:
OBSOLETE
use vtkwave_remove_glitches
```

vtk_wave_save

Obsolete

```
Usage:
vtk_wave_save waveGroupName waveFile

Description:
OBSOLETE: Use vtkwave_save instead
```

vtk_wave_statistics

Obsolete

```
Usage:
vtk_wave_statistics startTime endTime array

Description:
Fills array with statistics about the waves that have been loaded. Returns the number
of waves that have been analyzed.

OBSOLETE: Please use vtkwave_statistics instead.
```

vtk_wave_sum

Obsolete

Usage:
vtk_wave_sum resultRes res1 res2

Description:
resultRes: the name for to store the results

OBSOLETE: Use vtkwave_sum instead.

vtkwave_addinterval

Usage:
vtkwave_addinterval waveName startTs endTs value

Description:
Adds a value to the wave that is good for the specified time range.

Examples:
vtkwave_addinterval x\a \$t1 \$t2 0
vtkwave_addinterval x \$t1 \$t2 \$value ; ## Add an interval to all waves in wavegroup x.

vtkwave_addpoints

Usage:
vtkwave_addpoints waveName time-value-list

Description:
Add a list of time/value pairs to the wave. No simplification occurs.

Examples:
vtkwave_addpoints w\x [list \$now 2.1]
vtkwave_addpoints w\x [list \$ts1 2 \$ts2 3]

vtkwave_addstring

Usage:
vtkwave_addstring waveName timestamp string

Description:
Add a string at given timestamp. No simplification occurs. Examples:
vtkwave_addstring wx {\$now} {placement phase}

Example:
vtkwave_addstring w\x \$now "placement phase"

vtkwave_ceil

Usage:
vtkwave_ceil waveOrGroup

Description:
Round up each point in the wave to the next larger integer.

vtkwave_copy

Usage:
vtkwave_copy Destination Source

Description:
Copy a wave or wave group.

Examples:
vtkwave_copy x\b x\a; ### Copy wave x\a into x\b
vtkwave_copy x y; ### Copy whole wave group y into wave group x

vtkwave_crop

Usage:
vtkwave_crop waveOrGroup startTs endTs

Description:
Crop the wave between startTs and endTs.

vtkwave_derivative

Usage:

```
vtkwave_derivative derivativeWave originalWave step
```

Description:

Compute the derivative wave from an original wave. This is of course a rough approximation of a derivative, given that the waves are step functions. The derivative in this case is the average slope in the intervals defined by "step"

Examples:

```
vtkwave_derivative x\d x\f 10
```

vtkwave_destroy

Usage:

```
vtkwave_destroy waveNameOrGroup
```

Description:

Destroys an existing wave object.

Examples:

```
catch {vtkwave_destroy x\a}      ;# Destroy the wave "x\a"  
catch {vtkwave_destroy x}       ;# Destroy the wave group "x"
```

vtkwave_dump

Usage:

```
vtkwave_dump waveName
```

Description:

Get info about all points in a wave and return in raw format.

Examples:

```
set data [vtkwave_dump x\running]  
puts $data  
0123456789 1 0123456790 2 ...
```

vtkwave_exists

Usage:

```
vtkwave_exists waveName
```

Description:

Test the existence of a specific wave.

Examples:

```
if [vtkwave_exists x\a] {  
  ## do something  
}
```

Returns:

The procedure returns 1 if the wave exists and 0 otherwise.

vtkwave_floor

Usage:

```
vtkwave_floor waveOrGroup
```

Description:

Round down each point in the wave to the next larger integer.

vtkwave_get

Usage:

```
vtkwave_get waveName array
```

Description:

Get info about all points in a wave.

Examples:

```
vtkwave_get x\running a  
parray a  
a(count) 33  
a(label) "running"  
a(group) "x"  
a(0,ts) 12892892829  
a(0,v) 22.2  
...  
a(32,ts) 1289289888  
a(32,v) 1.2
```

vtkwave_get_json

Usage:

```
vtkwave_get_json waveName
```

Description:

Get info about all points in a wave and return in JSON array format.

```
Examples:
set data [vtkwave_get_json x\running]
puts $data
[[0123456789,1],[0123456790,2]...]
```

vtkwave_intervals

```
Usage:
vtkwave_intervals waveName t1 t2 type

Description:
Create a wave with one point of value 1.0 at the boundaries of the intervals
specified by 'type', where type is one of: yearly quarterly monthly biweekly weekly
daily hourly minutely 60s or an integer number of seconds.

Examples:
vtkwave_intervals w\intervals $startTs $endTs 10
vtkwave_intervals w\intervals $startTs $endTs hourly
vtkwave_intervals w\intervals $startTs $endTs daily;    ## Accounts for DST
vtkwave_intervals w\intervals $startTs $endTs monthly;
```

See Also

[vtkwave_workweek](#)

vtkwave_invert

```
Usage:
vtkwave_invert invertedWave originalWave

Description:
Create an inverted wave from original wave. The inverted wave has value 1 where the
original is 0, and 0 elsewhere.

Example:
vtkwave_invert x\inv x\a ; ## Compute invert of wave x\a
```

vtkwave_levels

```
Usage:
vtkwave_levels waveName startTs endTs separateLevelFlag

Description:
If separateLevelFlag is 1, then compute the time spent at each level, else compute
the time spent at or below each level. It is expected that waveName has only integer
levels.
```

```
Examples:  
vtkwave_levels x\r $t1 $t2 1
```

vtkwave_load

```
Usage:  
vtkwave_load waveGroupName waveFile [start [end]]  
  
Description:  
Load a wave from a file into a wave group. The waves are cropped from start to end,  
where by default start is 0 and end is 'now'. This procedure should probably be  
called vtkwavegroup_load
```

See Also

[vtkwave_save](#)

vtkwave_multiply

```
Usage:  
vtkwave_multiply waveNameOrGroup factor  
  
Description:  
Multiply wave or all waves in a wave group by factor. The product or products are  
stored in place.  
  
Examples:  
vtkwave_multiply w\a w\b;      # Multiply w\a by w\b with result in w\a  
vtkwave_multiply X w\b;        # Multiply all waves in group X w\b with result in X.
```

See Also

[vtkwave_op](#)

vtkwave_op

```
Usage:  
vtkwave_op operation resultWaveOrGroup operandWaveOrGroup  
  
Description:  
Perform an operation on the result and operand waves or wave groups. Operation is one  
of MUL ADD SUB MAX MIN  
  
Examples:  
vtkwave_op ADD x\acc x\a      ; ## Add wave x\a to x\acc, with result in x\acc
```

```
vtkwave_op MUL x\acc x\f      ; ## Multiply wave x\acc by wave x\f with result in x  
\acc  
vtkwave_op MAX x\M x\a       ; ## Take the max of x\M and x\a, with result in x\M  
vtkwave_op ADD ACC A         ; ## Add, wave by wave, all waves in A to all waves in  
ACC
```

vtkwave_print

Usage:
vtkwave_print waveName labelPrefix

Description:
Print all points in a wave to stdout. This procedure is mostly meant to be used for debugging.

Examples:
vtkwave_print x\a "before simplification"
vtkwave_simplify x\a
vtkwave_print x\a "after simplification"

vtkwave_reduce

Usage:
vtkwave_reduce waveOrGroup BinInterval <max | avg >

Description:
Reduce a wave or all waves in a group into the specified bins. The result is stored in place.

Examples:
vtkwave_intervals bin\h \$t1 \$t2 hourly; ## Create a hourly wave.
vtkwave_copy x\m x\a; ## Copy x\a into x\m
vtkwave_reduce x\m bin\h max; ## Compute the maximum of wave x\m in each
hour.
vtkwave_copy xAVG x; ## Copy group x into xAVG
vtkwave_reduce xAVG bin\h avg; ## Get hourly average of all waves in xAVG.

vtkwave_remove_glitches

Usage:
vtkwave_remove_glitches waveOrGroup maxGlitchWidth

Description:
Removes all features shorter than the specified width from the wave.

Examples:

```
vtkwave_remove_glitches x\a 20;  ## Remove glitches from wave x\a  
vtkwave_remove_glitches x 20;  ## Remove glitches from all waves in group x
```

vtkwave_save

Usage:
vtkwave_save waveGroupName waveFile

Description:
Save the entire waveGroup into the specified file. This procedure should probably be called vtkwavegroup_save.

See Also

[vtkwave_load](#)

vtkwave_scale

Usage:
vtkwave_scale waveNameOrGroup scalarFactor

Description:
Multiply the wave or the wave group by a scalar.

Example:
vtkwave_scale X\a 2.2

vtkwave_set_separator

Usage:
vtkwave_set_separator character

Description:
A very technical api, just for developers. Used to set the separator between the wave group name and the wave name. The default character is the back-slash ' '

vtkwave_simplify

Usage:
vtkwave_simplify waveOrGroupName

Description:

Simplify the representation of a wave or of all waves in a group. The simplification eliminates redundant points on the waves and leads to a more compact representation of the waves. Simplification happens automatically as a side effect of many operations.

Examples:

```
vtkwave_simplify x\a  
vtkwave_simplify x
```

vtkwave_statistics

Usage:

```
vtkwave_statistics waveNameOrWaveGroup startTime endTime resultArray
```

Description:

Fills resultArray with statistics about the waves that have been loaded. The statistics include: area avg max min min,ts max,ts begin,ts end,ts If the first argument is a specific wave, the procedure returns nothing. If the first argument is a wave group, the procedure returns the number of waves that have been analyzed.

Example:

```
vtkwave_statistics x\a $startTs $endTs waveStats  
parray waveStats  
waveStats(max) 22  
waveStats(min) 0  
waveStats(area) 2928917  
...
```

vtkwave_sum

Usage:

```
vtkwave_sum result addendum1 addendum2
```

Description:

This procedure sums waves or wavegroups. The result is stored into result.

Examples:

```
vtkwave_sum aaa\r aaa\a aaa\b  
vtkwave_sum rrr aaa bbb
```

See Also

[vtkwave_op](#)

vtkwave_support

Usage:

```
vtkwave_support supportWave originalWave
```

Description:

Create a support wave from original wave. The support wave has value 1 where the original is different from zero, and 0 (zero) elsewhere.

Example:

```
vtkwave_support x\support x\a ; ## Compute support of wave x\a
```

vtkwave_workweek

Usage:

```
vtkwave_workweek waveName t1 t2 dayStart dayFinish weekendDays
```

Description:

Compute a wave to represent a regular workweek. The wave takes daylight savings times into account. The first argument is the name of the resulting wave. The arguments t1 and t2 are the beginning and ending of the wave and it is expected that t2 is greater than t1. The dayStart represents the hour and minutes of the start of the work day, in the format HH:MM. Similarly dayFinish represents the end of the work day, also in the format HH:MM. The last argument specifies which days are considered weekend days.

Examples:

```
vtkwave_workweek ww\work $t1 $t2 08:30 17:30 "Sat Sun"  
vtkwave_workweek ww\work $t1 $t2 07:00 16:00 "Fri Sat"
```

See Also

[vtkwave_intervals](#)

vtkwavegroup_get

Usage:

```
vtkwavegroup_get waveGroupName resultArray [BinInterval [MaxFlag]]
```

Description:

Get information about a group of waves.

vtk_wavegroup

vtk_wavegroup_apply_validity_wave

No information for this procedure.

vtk_wavegroup_get

Obsolete

```
Usage:
vtk_wavegroup_get waveName resultArray [binInterval [binOnMax]]

Description:
Get info about a wavegroup.

OBSOLETE: Use vtkwavegroup_get instead
```

vtk_waveplot

vtk_waveplot_addwave

```
Usage:
vtk_waveplot_addwave waveGroupName [OPTIONS]

Description:
Where options are:

-color    color
-index    index
-yscale   coeff
-wave     waveGroupName (to figure out wave index)
-bin      binInterval
-thick    thickness
-style    line|linearea|area|bar|impulse|step
-solid    -- same as -style area
-max
```

vtk_waveplot_create

```
Usage:
vtk_waveplot_create LABEL [OPTIONS]

Description:
```

The `vtk_waveplot*` procedures allow the creation of one plot at a time.

Start with

1. `vtk_waveplot_create`,
2. `vtk_waveplot_addwave ...` (repeated)
3. `vtk_waveplot_creategif GIFFILE`

Where options are:

```
-xrange min max
-yrange min max
-refTs t      -- Reference time in X axis
-title title
-w width
-h height
-tmargintop
-bmarginbottom
-lmarginleft
-rmarginright
```

vtk_waveplot_creategif

Usage:

```
vtk_waveplot_creategif gifFile
```

Description:

Create the gif file for the current wave plot.

vtk_which

vtk_which

Usage:

```
vtk_which executable
```

Description:

Equivalent to Unix command `which`.

Looks for the executable in the directories specified by the environment variable `PATH`.

Examples:

```
vtk_which cp
/bin/cp
vtk_which badexecutable
```

Returns:

The path to the executable or the empty string if the executable cannot be found.

vtk_widget

Usage:
vtk_widget cmd ...

Description:
This procedure is not meant to be used by the casual user. cmd is one of:

info widgetlist -- Return a list of all the existing widgets.
info canvaslist -- Return a list of all the existing canvases.

vtk_write

vtk_write_log

Usage:
vtk_write_log message

Description:
Write message to stdout or the daily log if redirected.

Examples:
vtk_write_log "hello world" stdout

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

advanced REST usage [125](#)
all about Tcl [130](#)

B

built-in server metrics [43](#)

C

CLI commands overview [8](#)
client user authentication [110](#)
command line arguments [142](#)
connection keep-alive [127](#)
connection, show from command line [16](#)
control flow [138](#)
customization of node.cgi [15](#)

E

environment variables [55](#)
error message: "cannot accept more connections" [49](#)
error message: "too many notify clients for user ..." [48](#)
example application: job submit and list [120](#)
example application: REST 101 [116](#)
example application: REST 101 use key authentication [119](#)
example applications: job submit and list [116](#)
expected duration [54](#)
expressions [135](#)

F

failure codes [40](#)
field reference [39](#)
format strings [39](#)

G

get ready for REST [115](#)

H

HTML pages [98](#)
HTTPS security considerations [127](#)

I

I/O [140](#)

incompatible fields [33](#)
inner loop timers [46](#)
issuing REST requests from the Swagger web UI [123](#)

J

JWT access tokens [127](#)
JWT token allocation [128](#)

K

key REST concepts [112](#)

L

launch jobs with non-default options [120](#)
lists [134](#)

M

metrics naming [42](#)
metrics storage [42](#)
miscellaneous [46](#)

N

nc_info.py [126](#)
nc_list.py [116](#)
nc_run.py [116](#)
node fields [20](#)

O

operators [37](#)
other files used by VOV [97](#)

P

pattern matching [135](#)
plotting [45](#)
predicates, in selection rules [33](#)

R

raw HTTP/VOV interface [98](#)
request URLs [113](#)
resource path [112](#)
REST API [112](#)
REST request detailed documentation [121](#), [122](#)

S

- selection rules [33](#)
- selection rules operators [37](#)
- sequence of firing a job on a tasker [47](#)
- size specification [39](#)
- socket, testing [18](#)
- string manipulation [137](#)

T

- Tcl procedures - A [143](#)
- Tcl procedures - B [145](#)
- Tcl procedures - C [146](#)
- Tcl procedures - D [148](#)
- Tcl procedures - E - F [150](#)
- Tcl procedures - G [151](#)
- Tcl procedures - H [157](#)
- Tcl procedures - I [158](#)
- Tcl procedures - J [162](#)
- Tcl procedures - L - O [163](#)
- Tcl procedures - P [166](#)
- Tcl procedures - R [169](#)
- Tcl procedures - S [171](#)
- Tcl procedures - U [181](#)
- Tcl procedures - V [182](#)
- Tcl procedures - W [225](#)
- Tcl procedures - X [226](#)
- Tcl procedures - Y [178](#)
- Tcl procedures - Z [227](#)
- Tcl syntax [130](#)
- Tcl VOV extensions [142](#)
- test, socket [18](#)

U

- using vovselect [44](#)

V

- VOV [39](#)
- VOV metrics [42](#)
- VOV metrics command line interface [43](#)
- VOV properties [10](#)
- vov security keys [102](#)
- VOV Subsystem Reference Guide [7](#)
- vov_rest_v3.py Python library module [121](#)
- vov-project [18](#)
- VOVDIR [10](#)

VovId [15](#)
VovID [16](#), [17](#), [18](#)
vovjobresumer [10](#)
vovjobskip [10](#)
vovmemtime [18](#)
vovnotifyd [10](#)
vovprop [10](#), [15](#), [18](#)
vovresourced [10](#)
VOVRestV3 Python class description [121](#)
vovsecurity [108](#)
vovserver [10](#), [17](#), [18](#)
vovserverdir [17](#), [18](#)
vovset [39](#)
vovshowconnection [16](#)
vovtestsocket [18](#)
VovTrace [15](#)
VovUser [10](#)
vovwavepp [17](#)
VTK API available in vovtasker [227](#)
Vtk interface [45](#)
VTK procedures [229](#)
vtk_acl [229](#)
vtk_acl_op [229](#)
vtk_ae_getkeysfromfile [230](#)
vtk_alert [230](#)
vtk_alert_ack [230](#)
vtk_alert_add [231](#)
vtk_alert_forget [231](#)
vtk_alert_set_defaults [231](#)
vtk_annotation [231](#)
vtk_annotation_add [231](#)
vtk_annotation_count [232](#)
vtk_annotation_delete [232](#)
vtk_annotation_get [232](#)
vtk_annotation_modify [233](#)
vtk_artificial [233](#)
vtk_artificial_dependency_declare [233](#)
vtk_asynch [233](#)
vtk_asynch_notify_set_filter [233](#)
vtk_bar [234](#)
vtk_bar_chart [234](#)
vtk_base64 [234](#)
vtk_base64_decode [234](#)
vtk_base64_encode [235](#)
vtk_cache [235](#)
vtk_cache_clear [235](#)
vtk_chain [235](#)

vtk_chain_get [235](#)
vtk_checkout_checkin [236](#)
vtk_checkout_create [236](#)
vtk_checkout_delete [236](#)
vtk_checkout_get [237](#)
vtk_checkout_set [237](#)
vtk_checkouts [237](#)
vtk_checkouts_get [237](#)
vtk_client [238](#)
vtk_client_log_genkey [238](#)
vtk_client_log_getkey [238](#)
vtk_client_log_login [238](#)
vtk_client_mgr [238](#)
vtk_client_name [239](#)
vtk_client_version [239](#)
vtk_clients [239](#)
vtk_clients_get [239](#)
vtk_cmd [240](#)
vtk_cmd_quote_for_shell [240](#)
vtk_crc [240](#)
vtk_crc string [240](#)
vtk_dailyclose [240](#)
vtk_dailylog_open [241](#)
vtk_dailylog_write [241](#)
vtk_date [242](#)
vtk_date_format [242](#)
vtk_date_pp [242](#)
vtk_debug [242](#)
vtk_debug_set [243](#)
vtk_disable [243](#)
vtk_equivalence [244](#)
vtk_equivalence_get_cache [244](#)
vtk_equivalence_set_cache [245](#)
vtk_equivalences [245](#)
vtk_equivalences_debug [245](#)
vtk_equivalences_find [245](#)
vtk_equivalences_get [246](#)
vtk_equivalences_refresh [246](#)
vtk_equivalences_show [246](#)
vtk_event [247](#)
vtk_event_control [247](#)
vtk_event_generate [248](#)
vtk_event_get [248](#)
vtk_event_hurl [249](#)
vtk_event_subjects_map [249](#)
vtk_event_verbs_map [249](#)
vtk_eventfilter [250](#)

vtk_eventfilter_append [250](#)
vtk_eventfilter_clear [250](#)
vtk_eventmonitor [251](#)
vtk_eventwidget [251](#)
vtk_exclude [251](#)
vtk_exclude_rule [252](#)
vtk_exit [252](#)
vtk_feature [252](#)
vtk_feature_checkout [253](#)
vtk_feature_delete [253](#)
vtk_feature_find [253](#)
vtk_feature_get [253](#)
vtk_feature_get_or_create [254](#)
vtk_feature_get_version_pool [254](#)
vtk_feature_set [254](#)
vtk_feature_set_version_pool [255](#)
vtk_features [255](#)
vtk_features_get [255](#)
vtk_file [255](#)
vtk_file_get [255](#)
vtk_file_put [256](#)
vtk_file_read [256](#)
vtk_filesystem [256](#)
vtk_flexlm_monitor [256](#)
vtk_flexlm_monitor_all [257](#)
vtk_foreign_bucket_reset [258](#)
vtk_fs [258](#)
vtk_fs_latency [258](#)
vtk_fs_stat [259](#)
vtk_fsgroup [259](#)
vtk_fsgroup_create [259](#)
vtk_fsgroup_delete [260](#)
vtk_fsgroup_find [260](#)
vtk_fsgroup_get [260](#)
vtk_fsgroup_set [261](#)
vtk_fsgroup_update [261](#)
vtk_ftlm [261](#)
vtk_ftlm_agent_all [261](#)
vtk_ftlm_agent_define [262](#)
vtk_generate_ae_keypair [262](#)
vtk_generic [262](#)
vtk_generic_get [262](#)
vtk_geo [263](#)
vtk_geo_get [263](#)
vtk_geo_set [263](#)
vtk_get [263](#)
vtk_graph [264](#)

vtk_groupname [265](#)
vtk_gui [265](#)
vtk_gzip [266](#)
vtk_hash [266](#)
vtk_host [266](#)
vtk_host_forget [267](#)
vtk_host_id [267](#)
vtk_hostname [267](#)
vtk_init [268](#)
vtk_input [268](#)
vtk_input_declare [268](#)
vtk_input_declare_ex [269](#)
vtk_integer [269](#)
vtk_job [270](#)
vtk_jobclass [271](#)
vtk_jobclass_set_autoforget [271](#)
vtk_jobclass_set_idle_delays [10](#), [272](#)
vtk_jobclass_set_max_reschedule [272](#)
vtk_jobclass_set_preemption_method [272](#)
vtk_jobclass_set_revocation_delay [10](#), [272](#)
vtk_jobqueue [273](#)
vtk_jobqueue_delete [273](#)
vtk_jobqueue_get [273](#)
vtk_la_ctrl [274](#)
vtk_licdaemon [274](#)
vtk_licdaemon_delete [275](#)
vtk_licdaemon_find [275](#)
vtk_licdaemon_get [275](#)
vtk_licdaemon_get_or_create [275](#)
vtk_licdaemon_set [276](#)
vtk_licdaemon_update [276](#)
vtk_licdaemons [276](#)
vtk_licdaemons_find_by_tag [276](#)
vtk_licdaemons_get [276](#)
vtk_license [276](#)
vtk_licmon [277](#)
vtk_limits [277](#)
vtk_limits_get [277](#)
vtk_limits_set [278](#)
vtk_logname [278](#)
vtk_lsf [279](#)
vtk_metric_add [279](#)
vtk_metric_op [279](#)
vtk_microcode [280](#)
vtk_microcode_get [280](#)
vtk_microcode_set [280](#)
vtk_modify_fields [280](#)

vtk_mq_enable_priority_based_alloc [281](#)
vtk_mq_get_priority_based_alloc_percentage [281](#)
vtk_mq_set_priority_based_alloc_percentage [281](#)
vtk_multiqueue [282](#)
vtk_multiqueue_get [282](#)
vtk_multiqueue_get_matches [283](#)
vtk_multiqueue_set [283](#)
vtk_new_process_session [284](#)
vtk_node [284](#)
vtk_node_barrierinvalid_get [284](#)
vtk_node_cache [284](#)
vtk_node_change_status [285](#)
vtk_node_connectivity [285](#)
vtk_node_forget [285](#)
vtk_node_format [286](#)
vtk_node_get [286](#)
vtk_node_get_ios [287](#)
vtk_node_get_sets [287](#)
vtk_node_history [288](#)
vtk_node_impact [288](#)
vtk_node_status_logging [289](#)
vtk_node_why [289](#)
vtk_nodeeditor [290](#)
vtk_nodes [290](#)
vtk_nodestatus [291](#)
vtk_nodestatus_get_color [291](#)
vtk_nodestatus_get_mask [291](#)
vtk_nodestatus_get_ncname [291](#)
vtk_nodestatus_set_color [292](#)
vtk_nodeviewer [292](#)
vtk_object [292](#)
vtk_object_get [293](#)
vtk_object_get_fromcache [293](#)
vtk_output [293](#)
vtk_output_declare [293](#)
vtk_output_declare_ex [294](#)
vtk_path [295](#)
vtk_path_canonicalize [295](#)
vtk_path_expand [295](#)
vtk_path_flush_caches [296](#)
vtk_path_flush_nfs_cache [296](#)
vtk_path_get_relative [296](#)
vtk_path_is_excluded [296](#)
vtk_path_mk_symlink [297](#)
vtk_path_permissions [297](#)
vtk_path_print_caches [297](#)
vtk_path_realpath [297](#)

vtk_path_relativize [298](#)
vtk_percent [298](#)
vtk_pie [298](#)
vtk_ping [299](#)
vtk_place [299](#)
vtk_place_find [299](#)
vtk_place_get [299](#)
vtk_place_get_or_create [300](#)
vtk_place_list_find [301](#)
vtk_place_set [301](#)
vtk_place_update [301](#)
vtk_port [302](#)
vtk_ppid [302](#)
vtk_preemptrule [302](#)
vtk_preemptrule_delete [303](#)
vtk_preemptrule_delete_all [304](#)
vtk_preemptrule_find [304](#)
vtk_preemptrule_get [304](#)
vtk_preemptrule_modify [305](#)
vtk_preemptrule_create [302](#)
vtk_product [306](#)
vtk_product_get_info [306](#)
vtk_product_name [306](#)
vtk_prop [306](#)
vtk_prop_decr_and_get [306](#)
vtk_prop_delete [307](#)
vtk_prop_get [10, 307](#)
vtk_prop_incr_and_get [307](#)
vtk_prop_list [10, 308](#)
vtk_prop_set [10, 308](#)
vtk_protocol [308](#)
vtk_protocol_id [308](#)
vtk_protocol_info [309](#)
vtk_pty_server_open [309](#)
vtk_reservation [309](#)
vtk_reservation_create [310](#)
vtk_reservation_delete [311](#)
vtk_reservation_delete_orphans [311](#)
vtk_reservation_get [311](#)
vtk_reservation_update [311](#)
vtk_resource [312](#)
vtk_resource_delete [312](#)
vtk_resourcemap [312](#)
vtk_resourcemap_add_lm_handles [312](#)
vtk_resourcemap_add_nc_jobs [313](#)
vtk_resourcemap_change_grab [313](#)
vtk_resourcemap_clear_reservations [313](#)

vtk_resourcemap_delete [314](#)
vtk_resourcemap_forget [314](#)
vtk_resourcemap_get [315](#)
vtk_resourcemap_list [315](#)
vtk_resourcemap_reserve [315](#)
vtk_resourcemap_return [316](#)
vtk_resourcemap_set [316](#)
vtk_resourcemap_set_in_bulk [318](#)
vtk_resourcemap_set_limit [318](#)
vtk_resourcemap_set_limit_using_client [318](#)
vtk_resourcemap_set_revocation_delay [319](#)
vtk_resourcemap_steal [319](#)
vtk_resources [320](#)
vtk_resources_get [320](#)
vtk_resources_parse [320](#)
vtk_retrace [320](#)
vtk_retrace_check [321](#)
vtk_retrace_format [321](#)
vtk_retrace_get [321](#)
vtk_retrace_start [322](#)
vtk_retrace_start_job_or_set [323](#)
vtk_retrace_stop [324](#)
vtk_sanity [324](#)
vtk_select [325](#)
vtk_select_create [325](#)
vtk_select_delete [326](#)
vtk_select_get [327](#)
vtk_select_loop [327](#)
vtk_selectrule [328](#)
vtk_server [329](#)
vtk_server_config [49](#), [329](#)
vtk_server_dir [329](#)
vtk_server_kill [330](#)
vtk_server_log [330](#)
vtk_server_setenv [330](#)
vtk_server_unsetenv [331](#)
vtk_set [331](#)
vtk_set_cache [331](#)
vtk_set_cohort [332](#)
vtk_set_create [332](#)
vtk_set_find [333](#)
vtk_set_forget [333](#)
vtk_set_get [334](#)
vtk_set_get_elements [334](#)
vtk_set_get_elements_chunks [335](#)
vtk_set_get_elements_enhanced [335](#)
vtk_set_get_or_create [335](#)

vtk_set_operation [336](#)
vtk_set_recompute [336](#)
vtk_set_rename [336](#)
vtk_set_report [337](#)
vtk_set_set [337](#)
vtk_set_statistics [337](#)
vtk_set_subselect [338](#)
vtk_setbrowser [339](#)
vtk_sign [339](#)
vtk_signal [340](#)
vtk_strings [340](#)
vtk_substitute [340](#)
vtk_swd_get [341](#)
vtk_swd_set [341](#)
vtk_tasker [341](#)
vtk_tasker_attach_to_agent [342](#)
vtk_tasker_config [342](#)
vtk_tasker_create [343](#)
vtk_tasker_define [344](#)
vtk_tasker_delete [344](#)
vtk_tasker_find [345](#)
vtk_tasker_get [345](#)
vtk_tasker_get_colormap [346](#)
vtk_tasker_get_jobs [346](#)
vtk_tasker_modify [346](#)
vtk_tasker_nc [346](#)
vtk_tasker_reserve [348](#)
vtk_tasker_set_defaults [349](#)
vtk_tasker_set_timeleft [349](#)
vtk_tasker_vnc [349](#)
vtk_taskerlist [350](#)
vtk_taskerlists [350](#)
vtk_taskerset [350](#)
vtk_taskerset_get [350](#)
vtk_tcl [351](#)
vtk_tcl_global_eval [351](#)
vtk_time [351](#)
vtk_time_pp [351](#)
vtk_time_psp [351](#)
vtk_timestamp_pp [352](#), [352](#)
vtk_trace [352](#)
vtk_trace_checkall [352](#)
vtk_trace_dump [353](#)
vtk_trace_is_connected [353](#)
vtk_trace_list_places [353](#)
vtk_trace_list_sets [354](#)
vtk_trace_list_users [354](#)

vtk_trace_rename [354](#)
vtk_trace_reopen [355](#)
vtk_trace_save [355](#)
vtk_transition [355](#)
vtk_transition_add [356](#)
vtk_transition_add_to_trace [356](#)
vtk_transition_array [357](#)
vtk_transition_chown_to_me [358](#)
vtk_transition_color_control [358](#)
vtk_transition_create_resumerjob [358](#)
vtk_transition_dispatch [359](#)
vtk_transition_failcode_pp [359](#)
vtk_transition_find [359](#)
vtk_transition_fire [359](#)
vtk_transition_get [360](#)
vtk_transition_get_failcode_mask [361](#)
vtk_transition_get_input_names [361](#)
vtk_transition_get_io_names [361](#)
vtk_transition_get_or_create [362](#)
vtk_transition_get_output_names [362](#)
vtk_transition_operation [362](#)
vtk_transition_preempt [362](#)
vtk_transition_set [363](#)
vtk_transition_set_failcode [364](#)
vtk_transition_update_stats [365](#)
vtk_umask [365](#)
vtk_umask_get [365](#)
vtk_umask_set [365](#)
vtk_unixgroup [366](#)
vtk_unixgroup_get_users [366](#)
vtk_user [366](#)
vtk_user_find [366](#)
vtk_user_forget [367](#)
vtk_user_get [367](#)
vtk_user_get_shell [367](#)
vtk_user_modify [368](#)
vtk_user_security [368](#)
vtk_user_set [368](#)
vtk_user_set_http_password [369](#)
vtk_usergroup [369](#)
vtk_usergroup_add [369](#)
vtk_usergroup_contains [370](#)
vtk_usergroup_delete [370](#)
vtk_usergroup_get [370](#)
vtk_usergroup_getgroups [371](#)
vtk_usergroup_populate [371](#)
vtk_usergroup_remove [371](#)

vtk_untime [372](#)
vtk_validate_keyformat [372](#)
vtk_wave [372](#)
vtk_wave_addinterval [373](#)
vtk_wave_addpoints [373](#)
vtk_wave_crop [373](#)
vtk_wave_load [373](#)
vtk_wave_multiply [374](#)
vtk_wave_remove_glitches [374](#)
vtk_wave_save [374](#)
vtk_wave_statistics [374](#)
vtk_wave_sum [375](#)
vtk_wavgroup [384](#)
vtk_wavgroup_apply_validity_wave [385](#)
vtk_wavgroup_get [385](#)
vtk_waveplot [385](#)
vtk_waveplot_addwave [385](#)
vtk_waveplot_create [385](#)
vtk_waveplot_creatgif [386](#)
vtk_which [386](#)
vtk_widget [387](#)
vtk_write [387](#)
vtkwave_addinterval [375](#)
vtkwave_addpoints [375](#)
vtkwave_addstring [376](#)
vtkwave_ceil [376](#)
vtkwave_copy [376](#)
vtkwave_crop [376](#)
vtkwave_derivative [377](#)
vtkwave_destroy [377](#)
vtkwave_dump [377](#)
vtkwave_exists [377](#)
vtkwave_floor [378](#)
vtkwave_get [378](#)
vtkwave_get_json [378](#)
vtkwave_intervals [379](#)
vtkwave_invert [379](#)
vtkwave_levels [379](#)
vtkwave_load [380](#)
vtkwave_multiply [380](#)
vtkwave_op [380](#)
vtkwave_print [381](#)
vtkwave_reduce [381](#)
vtkwave_remove_glitches [381](#)
vtkwave_save [382](#)
vtkwave_scale [382](#)
vtkwave_set_separator [382](#)

vtkwave_simplify [382](#)
vtkwave_statistics [383](#)
vtkwave_sum [383](#)
vtkwave_support [384](#)
vtkwave_workweek [384](#)
vtkwavegroup_get [384](#)