



ALTAIR

ONLY FORWARD

VOV Subsystem 2025.1.2

Administrator Guide

Contents

VOV Subsystem Administrator Guide	5
Client/Server Architecture	6
VOV Projects	9
VOV Registry	14
List Projects in the Registry	15
Get Started	17
Enable CLI Access on UNIX	17
Enable CLI Access on Windows	20
User Interfaces	23
Use VOV Subsystem Help	24
Start and Configure the Server	26
Start the vovserver	26
Server Working Directory	28
Port Number	31
Advanced Control of the Product Ports	33
vovservermgr	35
Server Configuration	37
Web Server Configuration	38
VOV Protocol Summary	42
Time Tolerance	43
Client Limitation and Tuning	44
Safe Server Threads	46
Set the Range for VovId	47
Server Configuration Parameters	48
Troubleshooting the Server	93
Change the Project Name, Server Host, or User	94
Journals	97
Log Files	97
Start and Configure the Server	99
Autostart Directory	100
Run Periodic Tasks with vovliveness	101
Run Periodic Tasks with vovcrontab	102
Job Status Triggers	104
Sanity Check for vovserver	106
Connect to a Project	108
Connect to a Project via the Command Line	108
Alerts	108
Thin Clients	111
Troubleshooting: Cannot Enable Project	111
Security	112
Protection from DoS Attacks	117

VovUserGroups.....	118
Stop the Server.....	120
Shut Down a Project.....	120
Destroy a Project.....	121
Recovery and Redundancy.....	122
Failover Server Candidates.....	122
How vovserver Failover Works.....	123
Crash Recovery Mode.....	125
Files, Equivalences, Exclusions.....	127
Canonical and Logical File Names.....	127
Databases.....	128
The LINK Database.....	130
The JOBSTATUS Database.....	132
Define Equivalences for File Names.....	133
Equivalence Cache.....	135
Use vovequiv to Check Equivalences.....	136
Exclude Files From the Graph.....	136
Change the exclude.tcl File.....	139
Automatic Zipping and Unzipping Files.....	140
Taskers.....	142
The taskers.tcl File.....	143
vtk_tasker_define.....	144
vovtaskermgr.....	147
Tasker ID.....	152
Tasker Attributes.....	152
Tasker Configuration Parameters.....	156
Tasker Power.....	158
Autokill of Jobs by vovtasker.....	159
Tasker RAM Saturation.....	160
Connect a Single Tasker.....	161
Tasker Status.....	163
Monitor Taskers.....	164
Tasker Resources.....	165
The taskerClass.table File.....	167
Time-Variant Taskers.....	167
Predefined VovResources:: Procedures.....	169
Tasker Policies.....	172
Tasker Reservations.....	172
Tasker Owner.....	178
Taskers: Mixed Windows NT and UNIX Environment.....	179
Start a Remote UNIX TASKER Details.....	179
Stop Taskers.....	180
Taskers Running as Root: Security Implications.....	180
vtk_job_control.....	182
Interfaces to Accelerator, SGE, LSF, and Others.....	184
Connect to a Third Party Batch Processing System.....	184
Interface FlowTracer with Accelerator.....	186

Interfaces to Other Batch Processing Systems.....	190
Configuration File for vovagent.....	193
Subordinate Resources.....	195
Interface with LSF using taskerLSF.tcl.....	196
Interface to LSF with vovlsfd.....	197
Monitor LSF Jobs in the GUI.....	204
Interface with Altair Accelerator using vovelasticd.....	205
Advanced Tasker Topics.....	211
Hardware Resources.....	211
HOST_OF_jobId and HOST_OF_ANCECEDENT.....	215
Tasker: vtk_tasker_set_timeleft.....	216
Troubleshoot Taskers.....	217
vovtsd (Tasker Service Daemon).....	218
vovtsd on UNIX.....	220
Map Host Names.....	220
Resource Management.....	222
Resource Monitoring.....	224
Resource Map Set.....	224
Resource Mapping.....	225
Resource Reservation.....	227
Delete Resources.....	228
Rank the Resource Agents.....	228
Resource Daemon Configuration.....	230
vovresourcemgr.....	231
Policies and Resources.....	233
Add Resources.....	234
CHOICEMASK Resource.....	234
License-based Resources.....	235
Configuring Job and License Checkout Matching.....	239
IT Administrator Topics.....	241
SSH Setup.....	241
Network Conditioning.....	241
File System Offsets.....	242
NFS and VOV.....	243
Network Overhead.....	246
Using PAM (Pluggable Authentication Modules) for Browser Authentication.....	246
Attribute Value Stream (AVS) File Syntax.....	247
Known Issues.....	249
Health Monitoring and vovnotifyd.....	249
Limits for Objects and Strings.....	251
VovScope.....	254
Legal Notices	256
Intellectual Property Rights Notice.....	257
Technical Support.....	263
Index	264

VOV Subsystem Administrator Guide

1

This manual provides information about managing a VOV project, including starting the server and connecting computing resources such as computers and licenses.

This chapter covers the following:

- [Client/Server Architecture](#) (p. 6)
- [VOV Projects](#) (p. 9)
- [VOV Registry](#) (p. 14)
- [Get Started](#) (p. 17)
- [Start and Configure the Server](#) (p. 26)
- [Connect to a Project](#) (p. 108)
- [Stop the Server](#) (p. 120)
- [Recovery and Redundancy](#) (p. 122)
- [Files, Equivalences, Exclusions](#) (p. 127)
- [Taskers](#) (p. 142)
- [Tasker Policies](#) (p. 172)
- [Interfaces to Accelerator, SGE, LSF, and Others](#) (p. 184)
- [Advanced Tasker Topics](#) (p. 211)
- [Resource Management](#) (p. 222)
- [IT Administrator Topics](#) (p. 241)

The VOV Subsystem Administrator Guide is intended for use with Altair Accelerator products, including Monitor, Accelerator, and FlowTracer. All of these products have a similar software architecture with a "project" managed by a vovserver.

You should already have a valid installation of a VOV product. If that is not yet done, refer to the *Installation Guide*.

"VOV" is a design management system originally developed at UC-Berkeley. For references, refer to ACM or IEEE Xplore Digital Library .

Client/Server Architecture

All of Altair Accelerator's products are based on a client/server architecture to support concurrent activities, team coordination, distributed data management, and distributed processing. The program vovserver runs in the background as a service to implement the main product features. It is the server.

Other programs may run as background daemons to help vovserver provide specialized features or actions. The set of daemon programs form a working server system that may be referred to as the vovserver.

The [total number of clients](#) that can connect concurrently to vovserver depends on the maximum available "descriptors", which can reach up to thousands for modern hardware.

The vovserver manages the dependency graph of jobs and files that is the flow data, and is responsible for the scheduling and dispatching of jobs in the proper sequence. It also manages interaction with four types of clients:


Client Type	Description
Tools/Jobs	The ability of jobs to connect to the vovserver at runtime is a key element of the vovserver approach. Through a number of "integration techniques", programs in a job, such as compilers, simulators, UNIX utilities, renderers or other tools, can connect to the vovserver to declare their inputs and outputs, so that the vovserver can maintain the dependency graph and the flow data.
Users	The users can connect to the server to query and modify the design flow data by using Command Line Interface (CLI) commands, by using the desktop GUI console, by using the web application console via a web browser, or by running scripts that use the API commands. <div> Note: A browser can also be considered a client.</div>
Taskers	These are helper clients that run on remote machines to provide access to computing resources to run jobs.
Proxies	Provide the server with information about external databases. These clients are rarely used. All essential databases, such as file systems, are internally supported by the vovserver.



Figure 1: The vovserver can Interact with Various Types of Clients

A typical FlowTracer configuration looks like the following:

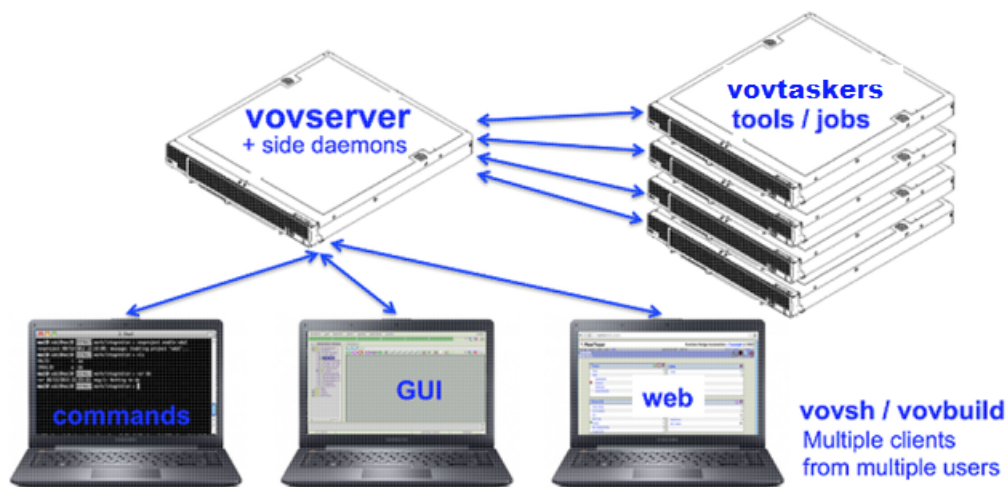


Figure 2:

The elements are:

- The main vovserver system that runs `vovserver` and other daemons, which manages the graph of dependencies, and jobs to schedule and run.
- The multiple vovtasker hosts that typically run on remote machines, which run the jobs and tools.
- Clients running on various machines, which provide access to the vovserver system by running a shell to call CLI programs, or running a desktop console GUI, or running a web browser to interact with the web console application of vovserver. These interact with vovserver to control the execution of job.

Relationship between FlowTracer, Accelerator, and Monitor

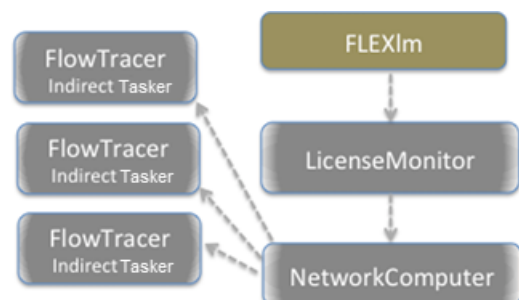


Figure 3: The Interaction of FlowTracer with Accelerator

Monitor and Accelerator are two other components of the VOV system. These components are not required to run FlowTracer, but are often used in conjunction with FlowTracer.

Both Monitor and Accelerator use the same architecture as FlowTracer. Monitor focuses on reporting of license utilization data, Accelerator focuses on efficient execution of jobs using available computing resources, while FlowTracer uses flow management techniques to submit a clean workload to the compute farm.

VOV Projects

A "project" is a named collection of jobs, together with all the files that are used and produced by those jobs. The contents of VOV projects can be custom organized.

Each design project is associated with one vovserver. The vovserver is started at the beginning of the project and normally runs until the project completes. Running in the background, the vovserver monitors the activity of the tools invoked by the designers, builds and manages the dependencies, services requests from the users, and dispatches jobs to the taskers.

Each VOV project may have its own private set of computing resources, or may interface to an existing compute farm, for example Accelerator or other schedulers.

Examples of projects:

- Compiling a program on one platform
- Compiling a program on N platforms
- Running regression for a software product
- Designing a chip or some sub-blocks of a chip
- Managing a workload (Accelerator)
- Managing the sampling of FlexNet Publisher daemons (Monitor)

VOV Projects Types

The following VOV projects are associated with Altair products.

Product	Default Project Name	Default Project Directory
Monitor	licmon	\$VOVDIR/../../licmon
Accelerator	vnc	\$VOVDIR/../../vnc
Accelerator Plus	wx	\$VOVDIR/../../wx
Allocator	la	\$VOVDIR/../../la
FlowTracer	-none-	-none-

VOV Project Server Host

The vovserver can run on a host that meets the following requirements:

- The host has access to all the design data, directly or through network file systems (such as NFS).

- The host has adequate main memory: least 64MB of RAM for graphs with 20,000 files in the graph
- each node in the graph uses about 1kB of RAM.

If the majority of the project files are stored in file systems residing on a single host (the file server), then the vovserver is best run on that host to avoid possible complications due to NFS or unsynchronized clocks. If internal policies or technical reasons discourage running the vovserver on the file server, then any other host can be used.

The combination of `host:port` must be unique among all running vovserver. It is not recommended to start servers with the same project name on different hosts; this leads to confusion for the users. However, running multiple servers with different names on the same host is an acceptable practice.

Create a VOV Project

1. To create a new VOV project for Accelerator, Monitor, Allocator, or Accelerator Plus, please use the appropriate manager script, as per the table below:

Product	Manager Command	Specialized Action
Monitor	Immgr	Checks that auxiliary daemons and scripts are up-to-date. The <code>vovproject</code> command does not check.
Accelerator	ncmgr	Checks that auxiliary daemons and scripts are up-to-date. The <code>vovproject</code> command does not check. Also creates the necessary entry in the <code>NC_CONFIG_DIR</code> (usually <code>\$VOVDIR/local/vncConfig</code>).
Accelerator Plus	wxmgr	Checks that auxiliary daemons and scripts are up-to-date. The <code>vovproject</code> command does not check. Also creates the necessary entry in the <code>NC_CONFIG_DIR</code> (usually <code>\$VOVDIR/local/vncConfig</code>).
Allocator	lamgr	Checks that auxiliary daemons and scripts are up-to-date. The <code>vovproject</code> command does not check.

2. To create a new VOV project for FlowTracer, use the `vovproject create` command.

```
% vovproject create -type <type> -port <port> -dir vovadmin <projectname>
```

Option

-type

Description

Select the project type which influences only the creation of a new project; the initialization of the [project files](#). These files control the project environment, tasker machines, users, security, etc., and are copied from the project type directory to the configuration files of the newly-created project.

The standard project types are usually augmented by the VOV site administrator to include ones specific to your site's work environment. This reduces the need for users to configure their own projects. The types are defined as subdirectories in `$VOVDIR/local/ProjectTypes`.

If no specific type is named, the new project will receive the template configuration files from the *generic* type. The following standard project types are available:


Project Type	Description
--------------	-------------

<i>generic</i>	The default type. Specialized for FlowTracer.
-----------------------	---

<i>licadm</i>	Specialized for an older form of Altair licensing based on vovserver, now replaced by ALM.
----------------------	--

More types can be added by creating new subdirectories and by modifying the template configuration files in the directory for that project type. Start a new project type by copying the configuration files from the `"$VOVDIR/./common/etc/ProjectTypes/generic"` type, and change them to suit the needs at your site.

For example, you might create a *Simulation* type, or a *PhysicalDesign* type, with specific tasker machines.

Option	Description
-port	<p>Choose a port number for the project. It is required that each project running on a host has a unique port; this is an operating system restriction. By default, the port number is computed automatically by hashing the project name into the range 6200-6255. It is possible for two project names to map to the same port. Such port collisions are handled by searching upward in the registry (starting at 7200) until an open port is found.</p> <div> Note: To run multiple projects concurrently on the same host, ensure they use different ports.</div>
-dir	<p>Choose the directory where you want to store all administration files for the project. The directory in which the vovserver is launched is called the <i>server working directory</i> (SWD). The server working directory contains the VOV database as well as all the project specific files. By default, the server working directory is <code>~/vov</code> on UNIX and <code>\$VOVDIR/swd</code> on Windows. For very important projects, the server working directory is explicitly set in an area where it can be subject to the same revision control methods as all the rest of the design data. In such cases, the directory is often called <code>vovadmin</code>.</p> <p>You also need to consider the canonical name of the Server Working Directory. For further information, refer to Canonical and Logical File Names.</p>
<projectname>	<p>Choose a name for the project. The project name can be any alphanumeric string, such as: <code>cpu</code>, <code>badge</code>, <code>MicroProcessor</code>, <code>mike</code>, <code>xyz99</code>. The name is case sensitive. Special characters, underscores <code>"_"</code> and dashes <code>"-"</code> are allowed.</p> <p>The name of the project and the host must each be 100 characters or less.</p> <p>The following strings are not allowed as project names: <code>"unknown"</code> <code>"none"</code>.</p>

Option

Description

The project name is used to compute the default **port number** used by the vovserver to listen for connections. See above for restrictions on the port number.

The following is an example of creating a project using the default port number, the default type *generic*, and the default working directory:

```
% vovproject create test
Creating a new project:
Directory /Users/designerabc/vov
Type      generic
Name      test
Port      automatic
vovproject 11/11/2015 07:28:30: message: Creating directory /Users/designerabc/
vov/test.swd/.
vovproject 11/11/2015 07:28:30: message: Created setup file '/Users/designerabc/
vov/test.swd/setup.tcl'
vovproject 11/11/2015 07:28:30: message: Copying initial DB config file...
vovproject 11/11/2015 07:28:36: message: Starting a VovServer
for project      test@mac05.local
PORT=automatic
vovserver Nov 11 07:28:36 VOV server MacOS/2015.11 Nov  5 2015 09:17:16
Copyright © 1995-2020, Altair Engineering
vovserver Nov 11 07:28:36 Version MacOS/2015.11 Nov  5 2015 08:55:57 Copyright ©
1995-2020,Altair Engineering
vovserver Nov 11 07:28:36 Project was started on: Thu Nov 11 07:28:36 2015
vovserver Nov 11 07:28:36 The server port is 6392
vovserver Nov 11 07:28:36 Running with capacity 256
vovserver Nov 11 07:28:36 ##### READY TO RECEIVE port=6392 #####
vovserver Nov 11 07:28:36 The server pid is 41613
vovserver Nov 11 07:28:36 << Redirecting output  /Users/designerabc/vov/test.swd/
logs/server.2015.11.11.log >>
```

VOV Registry

The VOV Registry contains the list of all projects known to an installation of Altair Accelerator software. The registry is used by the `vovproject` command, and stores metadata such as the host machine and port number of a project.

```
% vovproject list
PROJECT      OWNER  HOST      PORT  STATUS
1  IDT       johnak linux33   6425  stopped
2  TEM       johnak mac05   6430  stopped
3  chip1     johnak mac1ap 6250  running
4  chiptest  johnak mac05   6442  obsolete
```

The registry is normally in the directory `$VOVDIR/local/registry` but an alternative location may be specified by setting the environment variable `VOV_REGISTRY`.

The registry directory contains one entry per project. Each entry is a file with a name of the form `project@host`. The file is written in Tcl syntax and contains key information about the project. This is known as a *regentry* file.

As of version 2015.03, the registry places the regentry files for each user's projects under a subdirectory named for their user account. In the following example, the file would be in `$VOV_REGISTRY/cadmgr/vnc@rtda01`.

The `vovproject` command should automatically move a project's regentry file into the user-specific subdirectory the first time the project is started on 2015.03. To revert to an earlier release, you must move the regentry file manually.

For example, an entry called `vnc@rtda01` may contain the following information:

```
# -- This registry file is generated automatically. DO NOT EDIT!
# -- Copyright © 1995-2020, Altair Engineering
set host      "rtda01"
set project   "vnc"
set product   "nc"
set owner     "cadmgr"
set startdate 1331080351; # Tue Mar  6 16:32:31 2012
set version   "2015.03"
set protocol   "8.2.1"
set protocolid "120150304"
set swd        "/remote/proj/cadmgr/vnc"
set port      6271
set pid       3153
set description {}
set status     "running"
set vovdir     "/remote/release/VOV/2015.03_46113_Apr15/linux64"
set timestamp  1436228536; # Mon Jul  6 17:22:16 2015
set swdmap(keys) {unix canonical registry windows full }
set swdmap(unix)  {/remote/proj/cadmgr/vnc}
set swdmap(canonical)  {{VNCswd}}
set swdmap(registry)  {/remote/proj/cadmgr/vnc}
set swdmap(windows)  {r:/VOV/vnc}
set swdmap(full)    {/remote/proj/cadmgr/vnc}
```

The project's regentry file is initially created by the `vovproject create` command. Thereafter, the running `vovserver` updates its registry entry file every 10 to 30 minutes. During normal shutdown, the server updates the time stamp to the shutdown time and sets the `status` value to "stopped".

List Projects in the Registry

1. To list your project entries in the registry, use the following command:

```
% vovproject list
```

The above command will only list projects belonging to the user who issues the command.

2. To get detailed information about all projects, use this command:

```
% vovproject list -all -l
```



Note: If the server stops suddenly because it is killed or its host machine crashes, the registry entry will not reflect the correct status of the project. The `vovproject list` command will show 'obsolete' for projects whose status is `running` but whose time stamp is older than 1800 seconds (1/2 hour).

3. To determine whether a project's server is running, use the UNIX `ps (1)` command on the machine that hosts the project.

Registry Concerns

Because creating a VOV project also creates the registry entry, the registry directory must be on a writable file system with permissions permitting VOV users can create files there.

As the `vovserver` update their registry entry files periodically, the registry directory must be mountable from any machine which hosts a VOV project. The standard registry directory location satisfies this requirement as it is a subdirectory of the registry that contains the VOV software.



Note: A file system that is exported as read-only does not meet the requirements; it will not work.

Multiple versions of VOV can be set up to access the same set of projects by sharing the registry directory or the entire `local` directory among the versions. To set this up:

1. Move the local directory as a sibling to the version directories.
2. Create symbolic links from the local directory of each version to the shared directory.

Missing Registry Entry Files

Sometimes a project's regentry file may be absent or corrupted (for example, truncated to zero because the owner is over quota). Here there are two cases: vovserver running, and not running.

- If the vovserver is running it will re-write the registry file if possible.
- If the vovserver was not running, you can not start it using `vovproject`, because the regentry file that contained the project metadata is gone.

You can recreate the regentry file by starting vovserver manually as follows, starting vovserver from the command line.

```
+ determine the machine on which to start vovserver by  
  examining the project's setup.tcl file  
+ get a shell as the project owner on that machine  
+ change to the parent of the <project>.swd directory  
+ enter the project environment by  
% ves <project>.swd/setup.tcl  
+ start vovserver  
% vovserver -jsb $VOV_PROJECT_NAME
```


Get Started

Enable CLI Access on UNIX

This section explains how to set up a UNIX user's shell environment to have a proper context for the user to run installed Altair Accelerator programs from the command line. The programs that are run from the command line are called the CLI commands.

When a UNIX user logs into the system, a login script is executed which sets up their working environment. The default shell for the user determines what login script will run. A csh shell uses a `.cshrc` login script. A bash shell uses a `.profile` login script. The best way to set up a user's shell environment to run Altair Accelerator programs is to change the login script to properly set the user's working environment.

Altair Accelerator products require that the PATH environment variable be set to include the directories in the Altair Accelerator release which hold programs that will be run from the command line. Also, there are two environment variables that need to be set so that when an Altair Accelerator program is run, it will know where the release is installed and know what type machine it is running on. These two environment variables are VOVDIR and VOVARCH.

The Altair Accelerator product installation provides a helper file that can be sourced in order to set the needed environment variables to values that are appropriate for the local situation.

The intended use of the helper file is to have it sourced from a user's shell login script so that the user gets a proper environment without doing anything extra.

There is one helper file for each of the shell types that exist on UNIX. One helper script file is in the csh syntax and the other is in the shell syntax.

- vovrc.csh
- vovrc.sh

You can change each user's shell login script to source the file that is appropriate for the particular shell that they use.

Shell	Instructions
C-shell, tcsh	<p>Add the following line to your <code>.cshrc</code> csh login script file:</p> <pre>source /<install_path>/<version>/<platform>/etc/ vovrc.csh</pre>

Shell	Instructions
sh, ksh, bash, zsh	<p>Add the following line to your <code>.profile</code> shell login script file:</p> <pre>. /<install_path>/<version>/<platform>/etc/vovrc.sh</pre>

Verify Access to Altair Accelerator Products

After making the changes to source the helper files, and then logging in, you can check that needed environment variables are set properly by looking at environment variables to note that the `PATH` value contains a reference to the folders in the release which hold programs, and to note that `VOVDIR` and `VOVARCH` are set. `VOVDIR` should contain the path to the installation. `VOVARCH` should contain the name that matches the machine type it is running on.

1. At the command prompt, enter the following:

```
% echo $PATH
% echo $VOVDIR
% echo $VOVARCH
```

Beyond just looking, you can try running a Altair Accelerator product program as a positive test that the context is set up properly. One program that is simple to run and is available with all Altair Accelerator products is `vovarch`. This program reports on the machine type on which it is running. A side effect is that it tests that needed environment variables are set properly. If the environment is valid, the program will run and report the correct machine type. If the environment is not valid, the program will not work.

2. From the shell prompt, run the command `vovarch`.

```
% vovarch
```

You should get a response that is similar to one of the following, `"linux64"`, or `"macosx"`. This indicates the program was found, it ran, and it produced an expected output that matches your UNIX environment.

You have successfully set up the environment and verified it is correct.

3. If the response is `Command not found`, then the working environment does not have a `VOVARCH` setting for the programs in the Altair Accelerator products install area. If this is the case, review the steps to make sure the helper file from Altair Accelerator is being sourced correctly.

```
% vovarch
linux64
... or ...
macosx
```

or

```
% vovarch
```

```
vovarch: Command not found.
```

Enable Altair Accelerator for Non-Interactive Shells

It is possible to have the shell login script build a different working environment for interactive and non-interactive shells. The non-interactive environment is used when you run a batch script.

A common way this is done is to do an early exit from the login script file for non-interactive shells. For CSH, this can be done by testing for the existence of the shell variable "prompt".

Early exit from non-interactive shell login CSH script:

```
# This is a fragment of .cshrc.  
:  
# Batch scripts can skip doing actions needed by interactive scripts.  
if ( ! $?prompt ) then exit  
:  
# Below are actions needed by interactive scripts.  
:
```

If this exit happens early in the script, before sourcing the Altair Accelerator helper file, then the environment variables will not be set for the non-interactive shell.

A batch script needs access to Altair Accelerator products. This means that the non-interactive shell needs to have the needed environment variables set.

You should place the code that sources the helper file from Altair Accelerator early in the shell login script, before any logic causes the script to differ between interactive and batch shells.

Source helper file before exit in login CSH script:

```
# This is a fragment of .cshrc.  
:  
# Altair Accelerator env vars are needed by batch scripts.  
source /<install_path>/<version>/<platform>/vovrc.csh  
:  
# Batch scripts can skip doing actions needed by interactive scripts.  
if ( ! $?prompt ) then exit  
:  
# Below are actions needed by interactive scripts.  
:
```

Enable the Shell to Communicate With a Running Product Server

If you need to issue commands that will communicate to a running product instance, the instance will need to be "enabled", which involves setting certain environment variables that point the commands to the location of the running vovserver.

```
% vovproject enable instance-name
```

Some common instance names are "licmon" (for Monitor), "vnc" (for Accelerator), and "wx" (for Accelerator Plus). Instance names can be anything though since they are user-definable upon first start of each product.

Troubleshooting the UNIX Setup

An earlier section of this manual explained the importance of editing the `.cshrc` file so that batch shells would have the proper environment for running Altair Accelerator products.

The following is a command line to verify that the `.cshrc` file is set properly for batch shells. This runs the `vovarch` command within a batch context.

```
% csh -c vovarch
```

If this fails, the `.cshrc` file is not edited properly to enable access for batch shells. Review the details of the earlier topic on editing the `.cshrc` to enable access to batch shells.

Enable CLI Access on Windows

This section explains how to set up a Window user's command prompt environment to have the proper context for the user to run installed Altair Accelerator programs from the command line. The programs that are run from the command line are called the CLI commands.

Altair Accelerator products require that the `PATH` environment variable be set to include the directories in the Altair Accelerator release which hold programs that will be run from the command line. Also, there are two environment variables that need to be set so that when an Altair Accelerator program is run, it will know where the release is installed and know what type of machine it is running on. These two environment variables are `VOVDIR` and `VOVARCH`.

There are two methods for setting the correct environment. Both involve running the same context-setting bat script. This context-setting bat script establishes the correct environment variables for the local situation, reflecting where Altair Accelerator products are installed.



Note: This operation to set the environment is not required to use every Altair Accelerator feature on Windows. This operation is only needed to enable using the CLI commands from the command prompt.

Method 1: Use Windows Explorer to Set Command Line Environment

1. Using Windows Explorer, navigate to the Altair Accelerator installation directory.
2. Enter the `win64/startup` folder and double-click the `vovcmd.bat` script to run it.

This will open a command prompt with the proper environment settings for the Altair Accelerator and scripts to work.

When `vovcmd.bat` runs, it will execute the `win64/bat/vovinit.bat` script as part of what it does. The following section covering Method 2 explains what `win64/bat/vovinit.bat` does when it runs. It does the same thing when run by either method.

Method 2: Using Windows Command Prompt to Set Command Line Environment

1. In a command prompt window, navigate to the Altair Accelerator installation directory using the `cd` command.
2. Change directory to the `win64/bat` folder with `cd` and run the `vovinit.bat` script.
This will establish the needed environment for the open command prompt.

When `win64/bat/vovinit.bat` runs, it figures out needed environment variables and sets them, based upon where it is located. In particular, it sets `VOVDIR` to be the path to where Altair Accelerator is installed. It then executes another initialization script, `$VOVDIR/win64/local/vovinit.bat`, if it exists.

This is an initialization script that you can create and modify to perform site specific activities customized for your local configuration and usage. You can add commands to the `local/vovinit.bat` file that you want to run whenever a user starts up a command prompt.

After running both `vovinit.bat` scripts, the context of the command prompt has the correct environment needed so that CLI commands will work correctly with the installed Altair Accelerator.

Customize Actions Needed to Enable Access to Altair Accelerator Products on Windows

You can add commands to the `win64/local/vovinit.bat` file that perform specific operations that enable your Windows users to access Altair Accelerator programs properly, or to set up things on the machine to follow a standard convention.

You could add operations that perform actions such as these, and others:

- Mounting network drives
- Setting environment variables for local needs
- Establishing time synchronization

Example of a custom `$VOVDIR/win64/local/vovinit.bat`

```
rem -- Mount network drives:
rem -- In this example we mount the Altair Accelerator installation on drive v:
if not exist v:\nul net use v: \\somehost\altair

rem -- Set locally useful environment variables.
set VNCSWD=v:\vnc
set DLOG=d:\dailylog

rem -- Set Windows time from server on local network
```

```
rem -- Put this last; it may fail if lacking time set privilege
net time \\\timehost /set /y
```


Verify Context Is Working

When you have a command prompt open and expect that it has a context for accessing Altair Accelerator programs, check that the environment is set by looking at environment variables to note that the PATH value contains a reference to the folders in the release which hold programs, and to note that VOVDIR and VOVARCH are set. VOVDIR should contain the path to the installation. VOVARCH should contain the name that matches the machine it is running on.

Beyond just looking, you can try running a program as a positive test that the context is set up properly. One program that is simple to run and is available with all Altair Accelerator products is `vovarch`. This program reports on the machine type on which it is running. A side effect is that it tests that needed environment variables are set properly. If the environment is valid, the program will run and report the correct machine type. If the environment is not valid, the program will not work.

Run `vovarch` to verify the environment is set ok:

```
c:\ > vovarch
win64
```

 **Note:** The output will always show **win64** when running on any of the Microsoft Windows operating systems. This result is expected. It reports that we are on a generic "windows" architecture and indicates that the context is working.

If you are not able to verify that the context is valid, check the details within the `<installation_directory>/<version>/<platform>/bat/vovinit.bat` file.

Enable the Command Prompt to Communicate With a Running Product Server

If you need to issue commands that will communicate to a running product instance, the instance will need to be "enabled", which involves setting certain environment variables that point the commands to the location of the running vovserver.

```
c:\ > vovproject enable instance-name
```

Some common instance names are "licmon" (for Monitor), "vnc" (for Accelerator), and "wx" (for Accelerator Plus). Instance names can be anything though since they are user-definable upon first start of each product.

User Interfaces

You can access and manage the FlowTracer subsystem in multiple ways. This table explains the four different user interfaces. Mix and match which ones you use.

User Interface	Description	Features
Command Line Interface	The Command Line Interface is the set of programs you can run from the shell command line. These programs form a rich set of operations with command line parameters that provide detailed control of each task. Some tasks such as creating projects can only be done using a CLI command.	Enables automation of work. Commands can be run from a script.
Graphical User Interface	The Graphical User Interface is a custom application client using the client/server architecture. You use this program to have a clean, visual way of accessing the dependency graph, so you can control and manage the tracing of jobs. The graphical interface gives you a convenient way to do ad hoc activities for monitoring and controlling the processing.	Interactive. Offers full and immediate control.
Browser Interface	The browser interface is a web based application built into the FlowTracer system. The FlowTracer system provides the web server and web application as part of the standard installation. The server and application is accessed by a URL to the host. This web application provides a form based interface to access and manage the FlowTracer processing using the browser as your client program. The browser based web application includes links to the documentation.	Easy to use. Part of the normal install.
Tcl	Tcl is the language of choice for writing scripts to manage the FlowTracer system. The FlowTracer system provides a useful set of Tcl extensions that can be used by custom written Tcl scripts to access and manage processing. You can write Tcl scripts that have full control over FlowTracer processing, including the	Programmable, extensible. Most useful to advanced users.

User Interface	Description	Features
	development of new shell command line commands.	

Use VOV Subsystem Help

VOV Subsystem documentation is available in HTML and PDF format.

Access the Help when VOV Subsystem is Running

When VOV Subsystem is running, it displays the documentation through its browser interface. To access it from browser, you need to know which host and port VOV Subsystem is running on. Ask your administrator, or find the URL for VOV Subsystem with the following command:

```
% VOV Subsystem cmd vovbrowser  
http://comet:6271/project
```

In the example below, assume VOV Subsystem is running on host comet, port 6271. The URL for VOV Subsystem is:


```
http://comet:6271
```

To get the entire suite of Altair Accelerator documents, including FlowTracer™, Accelerator™, Monitor™ and the VOV subsystem, use the following URL:

```
http://comet:6271/doc/html/bookshelf/index.htm
```

Access the Help when VOV Subsystem is not Running

All the documentation files are in the Altair Accelerator install directory, so you can access them even if vovserver is not running. To do this, open `/installation_directory/common/doc/html/bookshelf/index.htm` in your browser.

 **Tip:** Bookmark the above URL for future reference.

Access the Help PDF Files

Altair Accelerator also provides PDF files for each of the guides. All the PDF files are in the directory `/installation_directory/common/doc/pdf`

Access the Help via the Command Line

The main commands of Accelerator are `nc` and `ncmgr`, with some subcommands and options. You can get usage help, descriptions and examples of the commands by running the command without any options, or with the `-h` option. For example,

```
% nc info -h  
nc:
```



```
nc:  NC INFO:
nc:  Get information about a specific job or list of jobs.
nc:  USAGE:
nc:  % nc info <jobId> [options]...
nc:  -h          -- Show this message
nc:  -l          -- Show the log file
nc:
```

Access the Help via the vovshow Command

Another source of live information is using the command `vovshow`. The following options are often useful:

<code>vovshow -env RX</code>	Displays the environment variables that match the regular expression <code>RX</code> provided.
<code>vovshow -fields</code>	Shows the fields known to the version of VOV in use.
<code>vovshow -failcodes</code>	Shows the table of known failure codes.

For example, to find a variable that controls the name of the stdout/stderr files, without knowing the exact name of that variable, the following command can be used:

```
% vovshow -env STD
VOV_STDOUT_SPEC          Control the names of file used to save stdout and
                          stderr. The value is computed by substituting
                          the substrings @OUT@ and @UNIQUE@ and @ID@.
                          Examples: % setenv VOV_STDOUT_SPEC
                          .std@OUT@.@UNIQUE@ % setenv VOV_STDOUT_SPEC
                          .std@OUT@.@ID@
```

The output provides a description of all the variables used by the FlowTracer system that include the substring "STD". In this example, the output result `VOV_STDOUT_SPEC`.

Start and Configure the Server

Start the vovserver

There are two ways to start the vovserver:

- `vovproject` is an easy to use interface to create, start, and stop a vovserver.
- `vovserver` is the binary executable.



Note: This file may be significant if a `vovproject` fails because of a corrupt registry entry.

vovproject

The `vovproject` tool is used to manage a VOV project.

An overview of setting up a new project:

1. Set up the server working directory with the necessary files. The default values for the files depend on the selected type. The list of available types is available in the directory `$VOVDIR/local/ProjectTypes`.
2. Start the server and the taskers for the project.

```
Copyright (C) 1995-2021,Altair Engineering    www.altair.com
```

DESCRIPTION:

```
The vovproject tool is used to manage a VOV project.
You can create new projects, start and stop
existing ones, enable the current shell to interact with a specific
project, etc...
```

```
If not otherwise specified, the tool assumes that the current project
is the one defined by the environment variables VOV_PROJECT_NAME,
VOV_HOST_NAME and VOV_PORT_NUMBER.
```

USAGE:

```
vovproject [command] [command arguments]
```

COMMANDS:

<code>archive</code>	# Archive projects.
<code>unarchive</code>	# Unarchive projects.
<code>backup</code>	# Make a backup copy of the trace
<code>create</code>	# Create a new project
<code>destroy</code>	# Completely remove all server data
<code>enable</code>	# Enable the shell to connect to the project
<code>info</code>	# Get info and stats about the project
<code>list</code>	# List all the projects
<code>rename</code>	# Change the name of the project

```
report          # Print reports about the projects
reread|reset|refresh # Re-read all configuration files
restore         # Restore the trace from a backup file
sanity          # Check internal consistency of server database
save           # Save trace database to disk
start           # Start a project server
stop           # Stop a project server
```

DETAILED HELP on above commands:
vovproject [command] -help

To create a new project, use `vovproject create`. To explicitly specify the working directory, use the option `-dir`. To explicitly specify the project type, use the option `-type`. Example:

```
% vovproject create [-dir serverdir] [-type type] [-port number] project
```

Connecting to the running server requires an enabled shell in which the environment variables `VOV_HOST_NAME` and `VOV_PROJECT_NAME` are properly set. To enable a shell, which is modifying an existing shell:

```
% vovproject enable project
```

If the project has already been created, the server can be started and stopped as follows:

```
% vovproject start project
% vovproject stop project
```

vovserver

```
usage: vovserver [-bctnl] [-p product] [-j] [-S project@host,port] [-P port]
                [-W port] [-E port] [-R port] [-x] [-k pubkey] <project>
-b:          Batch mode (UNIX only, silently ignored on NT)
-c:          Disable crash recovery
-t:          Start taskers automatically
-n:          Run as an Accelerator (NC) project (Obsolete: Use -p nc instead)
-l:          Run as a Monitor (LM) project (Obsolete: Use -p lm instead).
-p:          Set product type as one of: ft he la lm lms la nc rm wa wx (or
             auto)
-j:          Ignored. Supported for backwards compatibility
-S:          OLD: DO NOT USE. Start a server subordinate to the PrimaryServer
-P:          Force port number (in range [1000,65535])
-W:          Force web port number (0 to disable)
-E:          Force event port number (0 to disable)
-R:          Force readonly guest port number (0 to disable)
-x:          Run multi-threaded server
-k:          Specify the public API key of the owner on initial start.
<project>:  name of the project
```

Experienced VOV users can start the server directly by calling `vovserver`, provided that all project files are already available. Go to the server working directory and enter the following:

```
% cd ../<project_name>/swd
```

```
% ves project_name.swd/setup.tcl  
% ves BASE% vovserver -sb project_name
```

Server Working Directory

When a project is created, a server working directory (.swd directory) is created to hold configuration and state information about the project.

Location and Structure

The default location for the server working directory is within the `vovserverdir` directory, that is typically in the user's home directory, for instance, the `~/vov` directory.

During the project create command, it is possible to set the location for the project's server working directory to be a different place. It can be useful to place the project's server working directory within the project's top level directory. This allows for moving all the files needed for a project from one machine to another by taking files from a single branch of the file system.

The following list shows the structure of a project's server working directory (SWD), which contains the server configuration files for the project.

```
.../project.swd/equiv.tcl  
    /policy.tcl  
    /resources.tcl  
    /security.tcl  
    /setup.tcl  
    /taskers.tcl  
    /logs/  
    /...etc...
```

Find the SWD Default Location

From the command line, `vovserverdir` can be used to get information about the directory that is the default location for .swd directories. This command can also look for files within that file system branch.

```
% vovserverdir -physical  
/home/jjj/vov  
% vovserverdir -logical  
${HOME}/vov  
% vovserverdir -p setup.tcl  
/home/jjj/vov/project.swd/setup.tcl
```

Files in the Server Working Directory (.swd)

There are several files located in the project's server working directory.

File	Description
crontab.HOST	If it exists, this is the crontable for the project on a specific host. For information, refer to Run Periodic Tasks with vovcrontab .
equiv.tcl	This Tcl script defines the rules to generate logical names for files. Use <code>vovequiv -s</code> to check the effect of the file.
exclude.tcl	This Tcl script defines the rules to exclude files from the graph. All integrated tools use this file, while the vovserver ignores it.
policy.tcl	This is the main configuration file , which controls the behavior of the vovserver.
resources.tcl	This file describes the resources available for the project.
security.tcl	This file limits access to the vovserver to a defined set of users. This limitation is set up by specifying security rules .
setup.tcl	This Tcl script is used to set up the environment for a project. Typically, this script sets <code>VOV_HOST_NAME</code> and <code>VOV_PROJECT_NAME</code> and other environment variable(s) that are specific to the project.
taskers.tcl	This file describes the taskers to be connected to the vovserver. The tool <code>vovtaskermgr</code> uses this file.


Any change to a configuration file becomes visible to the vovserver by executing the following command:

```
% vovproject reread
```

Server Working Directory Subdirectories

The server working directory contains several subdirectories.

Subdirectory	Description
db/	Contains the SQL configuration file.
equiv.caches/	Contains preprocessed information derived from the <code>equiv.tcl</code> file. This is done because such execution may be time consuming. The result, which depends on the host, is cached in this directory.

Subdirectory	Description
	<div> Note: The files in this directory should not be modified by the user.</div>
environments/	<p>A location to store project specific environments. To do so, add a line to the <code>setup.tcl</code> file, similar to the following:</p> <div><pre>setenv VOV_ENV_DIR ...somewhere.../ \$env(VOV_PROJECT_NAME).swd/environments</pre></div>
jobs/	Used by Accelerator to store the logs for all completed jobs.
journals/	Contains the journals for the project. Each journal is a chronological list of all macro events occurred in the design. It is the project leader's responsibility to eliminate old issues of the journals. Recent issues are useful for debugging purposes.
logs/	Contains various logs. The server log is located in <code>server.YYYY.MM.DD.log</code> . The messages from the taskers are recorded in files with a name as follows: <code>tasker.<host>.N.M</code> . The file <code>daily.log</code> contains the activities done periodically each day. The files <code>resources.YYYY.MM.DD.log</code> and <code>tools.YYYY.MM.DD.log</code> are created by the server.
logs/checkouts	Used by Monitor to store the logs of all completed checkouts.
saved/	Contains compressed copies of the trace database. Use <code>vovproject backup</code> to save a copy; use <code>vovproject restore</code> if the main trace is corrupted.
scripts/	This directory holds scripts that related to the project, such as the scripts used for the periodic activities .
sq3/	Deprecated as of version 2015.09, the default directory for the SQLite database file.
trace.db/	Contains the trace database in ASCII and compressed format. <ul style="list-style-type: none">• <code>pr</code>: The Persistent Representation of the trace, in ASCII format.• <code>pr.gz</code>: The compressed trace.• <code>cr</code>: The Crash-Recovery files.

There are also subdirectories for each of the daemons that can be connected to the vovserver. Following is a list of some of the directories:

Subdirectory	Description
<code>vovpreemptd/</code>	Accelerator
<code>vovnotifyd/</code>	Monitor and Accelerator

Server Working Directory Name

To avoid issues such as looping, exercise caution with the name of the server working directory.

The SWD can be obtained with the utility `vovserverdir`. For example, the full path can be obtained with:

```
% vovserverdir
/user/john/myproject/vovadmin
```

The logical canonical name can be obtained with:

```
% vovserverdir -logical
${TOP}/vovadmin
```

A "circular loop" problem can occur with the name of the SWD, which can be prevented:

- To compute the full path to the SWD, either parse the setup file or know the rules in the equivalence file.
- To find either the setup file or the equivalence file, the full path to the SWD must be known.

To prevent issues from occurring, using the procedure `vtk_swd_get` in the `policy.tcl` file is strongly recommended.

```
# Example of using vtk_set_swd in the policy.tcl file
vtk_set_swd windows "x:/release/admin"
vtk_set_swd unix      /remote/release/rtda/admin
vtk_set_swd unix1     /net/zeus/release/rtda/admin
```

Another method to avoid accessing the server working directory is with a "thin client", which is described in the following section.

Port Number


The project name is normally used to compute the number of the TCP port on which the `vovserver` listens for connections. This is done by hashing the project name into the port range 6200-6455, using the function `vtk_port_number`.

Clients can connect to the `vovserver` and interact using either the proprietary VOV protocol or HTTP. It is also possible to connect to a specific "Web" port using either [HTTPS](#) or [HTTP](#) or to a special read-only port also using HTTP, all of which is explained in [Advanced Control of the Product Ports](#). However, a `vovserver` may use other than the default port as computed above, when the port is chosen manually at

project startup using the `-port` option of the `vovproject create` command, or when there is a conflict with another project that is already using the default port.

VOV normally maps the project name into an integer that ranges from 6200 to 6455 - unless otherwise directed by the environment variable `VOV_PORT_NUMBER`. This variable is typically set in the `setup.tcl` file.

The following table shows the mapping for some of the standard project names.

 **Note:** `licmon`, the project name for Monitor, uses a non-default port.

Project Name	Port number
intro	6244
licadm	6306
licmon	5555
vnc	6271
wa	6416

To find out a port number that can be used for a project, use `vovbrowser`, or `vsi` in a shell where the project is enabled.

```
% vovproject enable intro
venus intro@mercury [DEFAULT] 201> vovbrowser
http://mercury:6444/project
```

From Tcl, use `vtk_port_number` to compute the default port number for a given project. Example:

```
% vovsh -x 'puts [vtk_port_number vovtutorial]'
6407
```

The option `-checkenv` in `vtk_port_number` checks the value of the environment variable `VOV_PORT_NUMBER`.

```
% env VOV_PORT_NUMBER=7777 vovsh -x 'puts [vtk_port_number -checkenv vovtutorial]'
7777
% env VOV_PORT_NUMBER=7777 vovsh -x 'puts [vtk_port_number vovtutorial]'
6407
```

Conflicts in the Port Number

It is possible for different names to map to the same port number, thus making it impossible to run the corresponding servers concurrently on the same host. This occurs because only one process may listen on a host:port at a time, an operating system (OS) restriction.

If this problem occurs, it can be resolved as follows: change the host, change the project name, or explicitly choose the number of the port using the environment variable `VOV_PORT_NUMBER`. This variable is defined in the `setup.tcl` file in the server configuration directory.

The following subsection describes another method to resolve this issue.

Getting the Port Number in vovsh Connected to vovserver

If writing a script using `vovsh`, and finding the port number on which your script connected to vovserver is needed, the port number can be located by using `vtk_generic_get`. The following example assumes Monitor is owned by "cadmgr" and is using the regular port, 5555.

```
% vovproject enable -u cadmgr licmon% vovsh -x 'vtk_generic_get project P; puts  
$P(port) '  
5555
```

This method is recommended, as it queries vovserver for the port number.

From version 2014.11, `vtk_port_number` has the option `-checkenv`, which causes it to return the value of the environment variable `VOV_PORT_NUMBER`.

Advanced Control of the Product Ports

Enable a vovserver Project for Command Line Control

The commands described in this chapter require using a "vovproject-enabled shell". This is a command shell that has been configured to work with a specific vovserver. For more information, see [Enable CLI Access on UNIX](#) or [Enable CLI Access on Windows](#).

Change the Primary Port on a Running vovserver

The primary port cannot be changed without stopping and restarting the vovserver.

Change the Web Port on a Running vovserver

The web port can be changed dynamically with:

```
% vovsh -x 'VovServerConfig webport 6271'
```

Additional settings related to HTTPS can be set in `policy.tcl`:

```
# Example modification to policy.tcl  
set config(ssl.enable) 1  
set config(ssl.enableRawUrlOnHttp) 1
```

The vovserver must be notified of changes to `policy.tcl`:

```
% vovproject reread
```

Change the Read-only Port on a Running vovserver

The read-only port, also called the guest access port, can be changed dynamically with:

```
% vovsh -x 'VovServerConfig readonlyport 8283'
```

To maintain backwards compatibility, the read-only port can also be set in `policy.tcl`:

```
# Example modification to policy.tcl  
set config(readonly) 8283
```

The vovserver must be notified of changes to `policy.tcl`:

```
% vovproject reread
```

Recommendations for VOV_PORT_NUMBER for Monitor

- Set VOV_WEB_PORT_NUMBER to the well known port 5555
- Set the VOV_PORT_NUMBER to "any"

```
# In licmon.swd/setup.tcl
setenv VOV_PORT_NUMBER any
setenv VOV_WEB_PORT_NUMBER 5555
```

Recommendations for VOV_PORT_NUMBER for Accelerator

The following is a summary of the important recommendations when setting the VOV_PORT_NUMBER for Accelerator:

- It is important for Accelerator to restart quickly. To do so, avoid conflicts on the port used by the Accelerator vovserver.
- Keep the port for the web pages in the same place even if the vovserver uses a different port after a restart.
- Ensure the web port is different from all other ports. The easiest to do this is to assign the web port the lowest number from the internal list, then increment the value for other ports that you configure.

The following settings for VOV_PORT_NUMBER are recommended:

- Select a series of consecutive values from the internal port list, such as 6271 -6275.
- Set VOV_WEB_PORT_NUMBER to a known port:

For example:

```
setenv VOV_WEB_PORT_NUMBER 6271
```

- If you wish to have anonymous access via HTTP, set VOV_READONLY_PORT_NUMBER to some other fixed value. As mentioned above, our recommendation is +1 above the web port: 6272.
- Set VOV_PORT_NUMBER to a list of 3 or 4 known ports. As mentioned above, select the port numbers by continuing to increment from the value used for the web port.

For example:

```
setenv VOV_PORT_NUMBER 6273, 6274, 6275
```

```
# In setup.tcl for an Accelerator project
setenv VOV_WEB_PORT_NUMBER 6271
setenv VOV_READONLY_PORT_NUMBER 6272
setenv VOV_PORT_NUMBER 6273:6274:6275
```

Recommendations for VOV_PORT_NUMBER for FlowTracer

If you are going to have tens or hundreds of vovservers running on the same small number of machines, then it is most important to avoid conflicts on the ports, a goal that can be achieved by setting the value of VOV_PORT_NUMBER to "any" so vovserver can choose a port.

```
# In setup.tcl for an FlowTracer project
setenv VOV_PORT_NUMBER any
```

Running Multiple Product Instances on the Same Machine

Although not recommended, it is possible to run multiple instances of the same product, such as Monitor or Accelerator, on the same machine. In such cases, it is imperative that you carefully control the allocation of the ports to each instance, in order to avoid collisions (i.e. one of the instances is unable to start) and user confusion.

In particular, you have to make sure that the `setup.tcl` file for each instance in the `NC_CONFIG_DIR` directory (usually `$VOVDIR/local/vncConfig/<queueName>.tcl` for Accelerator) defines the following environment variables:

- **VOV_PORT_NUMBER:** Required. This is the main port and it **MUST** be different for each instance
- **VOV_WEB_PORT_NUMBER:** If you are using nginx as a front-end for the vovserver, you must assign a distinct value to each instance. Otherwise, you can set this variable to 0 to disable nginx.
- **VOV_READONLY_PORT_NUMBER:** This is an optional port, which is normally activated for both Monitor and Accelerator. Set this to 0 if you do not need to activate it, or choose a unique value for each instance.

vovservermgr

The `vovservermgr` command is added for system administrators. The `vovservermgr` command is the easiest way to modify configuration parameter settings, environment variable settings, and certain other features in the running vovserver. This command acts relative to the VOV-project enabled in the shell where it is issued.

This command has several subcommands, which provide an easier way to set vovserver configuration and environment variables other than `vtk_server_config`, et al. Other subcommands interface to memory chunking and scheduler tuning controls.

VovScope

VovScope provides insights into events (system call, network communications) happening at a particular instant while VOV is running. It is primarily meant for debugging, and can slow down the vovserver significantly.

Start and stop the VovScope process with the following processes:

```
Start:
vovservermgr configure set_debug_flag VovScope

Stop:
vovservermgr configure reset_debug_flag VovScope
```

- VovScope currently logs information about the primary server.
- VovScope logs information about all clients (FDs), including `vovresourced` and `vovdbd`. Anything that is returned by `vovshow -clients` can be read.
- `vovserver` receives events, while VovScope logs the events that VovScope is currently processing. For example, event: `GetProperty` (event code 189).
- Currently all the bytes written to FD, are reported by VovScope as well. Some of the bytes may or may not be printable.

The profiling records are written in the logs folder under the server working directory. The profiling records have filenames that start with prefix `vovscope`.

vovservermgr

vovservermgr: Usage Message

DESCRIPTION:

Modify settings and modes in running vovserver.

USAGE:

`vovservermgr <COMMAND> [OPTIONS] [command arguments]`

COMMANDS:

`config <Param> <Value>[-|+]`

Send a request to the server to set the configuration parameter `Param` to `Value`. Similar to the `vovsh vtk_server_config` command.

If `Value` is numeric, appending `-` prevents the request if the current value is already less than `Value` or appending `+` prevents the request if the current value is already greater than `Value`.

In all cases, minimum, maximum, or other validation constraints may be applied at the server depending on the parameter.

The token "DEFAULT" may be used for `Value` to specify the server default

value.

`setenv <Var> <Value>`

Set vovserver environment variable `Var` to `Value`. Similar to the `vovsh vtk_server_setenv` command.

`unsetenv <Var>`

Unset vovserver environment variable `Var`. Similar to the `vovsh vtk_server_unsetenv` command.

`mem`

Same as the `reducememory` command.

`reducememory`

Hint to vovserver to perform memory allocation bookkeeping and optimizations now.

`scheduler <suspend|resume|unsuspend|request|quick|update>`

Equivalent to `vovservermgr config scheduler <subcommand>`

`suspend` - the scheduler stops dispatching jobs

`resume` - the scheduler resumes normal operation

`unsuspend` - same as `resume`

`quick` - request scheduler in quick mode

`update` - request scheduler in update mode (rebuild buckets)

`request` - same as `update`

OPTIONS:

`-h` -- Show usage message.

`-v` -- Increase verbosity.

EXAMPLES:

`% vovservermgr reducemem`

`% vovservermgr setenv VOV_HOST_HTTP_NAME vncsrv.internal.mycompany.com`


```
% vovservermgr unsetenv VOV_HOST_HTTP_NAME
% vovservermgr scheduler suspend
% vovservermgr scheduler resume
% vovservermgr config maxNotifyClients 1000+
% vovservermgr config usepoll 1
```

SEE ALSO:

```
vovshow -policy
vtk_server_config
vtk_server_setenv
vtk_server_unsetenv
```

Server Configuration

vovserver configuration parameter values may be changed in a running vovserver using the CLI, or prior to the starting the server via the `policy.tcl` file. An administrator can configure the parameters in the running vovserver using the `vovservermgr` command or the `vtk_server_config` procedure.

 **Note:** In usage, all commands and parameters are case insensitive.

Using `vovservermgr`:


```
% nc cmd vovservermgr config PARAMETER_NAME PARAMETER_VALUE
```

Using `vtk_server_config`:

```
% nc cmd vovsh -x 'vtk_server_config PARAMETER_NAME PARAMETER_VALUE'
```

Example of a configuration:

```
% nc cmd vovsh -x 'vtk_server_config timeTolerance 4'
% nc cmd vovsh -x 'vtk_server_config timeTolerance 4'
```

 **Note:** A complete list of the current server configuration parameters is provided in the Server Configuration page.

Server configuration can be controlled by setting variables in the `policy.tcl` file. The variable can be set directly in the "config()" associative array, but it is best to set them with the procedure `VovServerConfig` as in:

```
VovServerConfig VARNAME VALUE
```

In either case, the name of the parameter `VARNAME` is case insensitive.

Example

The following is part of the `policy.tcl` file.

```
# This is part of the policy.tcl file.

# Example of parameters set with the procedure VovServerConfig
```

```
VovServerConfig readonlyPort 7111
VovServerConfig httpSecure 1

# Example of parameters set by assignment to array config().
set config(timeTolerance) 0

set config(maxBufferSize) 16000000
set config(maxNotifyBufferSize) 400000
set config(maxNotifyClients) 40;
set config(maxNormalClients) 400;

set config(maxAgeRecentJobs) 60;
set config(saveToDiskPeriod) 2h;
set config(autoShutdown) 2w; # Shut down after 2 weeks of inactivity.
set config(autoLogout) 1h; # Logout from browser interface.
set config(netInfo) 0; # Do not collect net information (fs,
hosts)

# Used by Accelerator for autoforget.
set config(autoForgetValid) 1h
set config(autoForgetFailed) 2d
set config(autoForgetOthers) 2d

set config(autoRescheduleThreshold) 2s

set config(preemptionPeriod) 3s
```

Below is an example of parameters set with the procedure VovServerConfig:

```
VovServerConfig readonlyPort 7111
VovServerConfig httpSecure 1
```

Web Server Configuration

HTTP Access Models

There are 3 HTTP access models:

- Legacy
- Internal/External
- Nginx

Legacy Webserver

The Legacy webserver is the basic web server that is internal to vovserver and serves content directly to web browser clients.

All traffic is transmitted using HTTP protocol and is unsecured. This method is appropriate for REST versions up to version 2.0.

This is the case when

- webport=0, or

- `webport != 0` and `webprovider=nginx`

Internal Webserver

The Internal webserver is an enhanced web server that is internal to vovserver, for secure pages and all REST versions.

The Internal webserver is established when

- `webport != 0` and `webprovider=internal`

To specify the web port at product start, refer to the product-specific documentation for startup. To change the port in an already-running product instance, see *Advanced Control of the Product Ports*. To enable SSL support (HTTPS), follow the steps in [Configure the TLS/SSL Protocol](#).

You get REST v3 API support from this webserver, and we still transparently delegate some HTTP requests to the old web server on the VOV port.

The Internal server securely handles all incoming traffic, decrypting it before handing it off to the locally running vovserver. Likewise, any response that is sent back to the browser is routed through the Internal webserver, which encrypts the response and sends it to the browser. This implementation is known as an SSL termination proxy.

nginx Webserver

The vovserver serves content to a proxy webserver (nginx), which communicates to web browser clients. Under this model, SSL can be enabled, securing all traffic using the HTTPS protocol.

The nginx web server is enabled when the web port is configured with a non-zero value. To specify the web port at product start, refer to the product-specific documentation for startup. To change the port in an already-running product instance, see *Advanced Control of the Product Ports*. To enable SSL support (HTTPS), follow the steps in [Configure the TLS/SSL Protocol](#).

For experts only, advanced customizations to the nginx configuration can be made by modifying its configuration template. Configuration templates are searched for in the following locations:

Order	Type	Path
1.	Instance-specific	<code>\$SWD/vovnginxd/conf/nginx.conf.template</code>
2.	Site-wide	<code>\$VOVDIR/local/config/vovnginxd/nginx.conf.template</code>

Order	Type	Path
3.	Installation-specific(edits not recommended)	\$VOVDIR/etc/ config/vovnginxd/ nginx.conf.template

If customizations are intended, it is recommended to start with a copy of the default configuration template shown at location 3 above and place into either location 1 or 2.



Note:

- The configuration template is copied into the nginx configuration directory located at `$SWD/vovnginxd/conf`, named as `nginx.conf`. The copy is made upon product start, as well as any time the web port or SSL configuration is changed.
- Changes to the actual configuration file can be read into nginx via the `vovdaemonmgr reread vovnginxd` command, but such changes will be overwritten the next time the configuration template is copied.
- The configuration template contains keywords surrounded by @ signs, such as `@WEBPORT@`, that are dynamically substituted with values during the copy process. Removal of these keywords is not recommended, as it may effect the ability for nginx to be reconfigured in the event of a vovserver failover.

Configure the TLS/SSL Protocol

The internal and nginx webserver support TLS/SSL Protocol communication via "https" - prefixed URLs when configured correctly.



Note: TLS encryption is enabled for client communication to the VOV port by default starting in 2025.1.0. TLS can be disabled by setting the new `comm.tls.enable` server configuration parameter to 0. Older version clients that do not use TLS will be allowed to connect without TLS encryption.

The vovserver serves content to a proxy webserver (nginx), which communicates to web browser clients. Under this model, SSL can be enabled, securing all traffic using the HTTP protocol.

When SSL is enabled, nginx will look for an SSL certificate/key pair in the following locations:

Order	Type	Path	Files
1.	Site-wide wildcard	\$VOVDIR/local/ssl	wildcard-crt.pem wildcard-key.pem
2.	Host-specific	\$SWD/config/ssl	hostname-crt.pem

Order	Type	Path	Files
			hostname-key.pem
3.	Host-specific (auto-generated and self-signed)	\$SWD/config/ssl	hostname-self-crt.pem hostname-self-key.pem



Note:

- For hostname, use the actual host name that will be used to access the web UI. This will be the value of VOV_HOST_HTTP_NAME that was set in the configuration. If not defined, the value of VOV_HOST_NAME is used instead.
To use the fully qualified domain name, the value of VOV_HOST_HTTP_NAME must be set.
- Self-signed certificates will present security warnings in most browsers.

Updating the TLS/SSL cert requires restarting the webserver so that the cert files can be re-read. For the internal webserver, see, "Restarting the Webserver" below.

Guest Access Port

The vovserver can be configured to enable a guest-access port, also called the read-only port due to the limited privileges allowed by the port. This port bypasses the login prompt and provides the user with a READONLY security principle, which disallows access to writable actions as well as certain pages in the UI.

To specify the guest access port at product start, refer to the product-specific documentation for startup. To change the port in an already-running product instance, follow the steps in *Advanced Control of the Product Ports*.

Transition from nginx Webserver to Internal

To transition from external (nginx) to the internal web server, follow these steps:

1. Shut down nginx with the command `vovdaemonmgr stop vovnginxd`.
2. Delay for 5 seconds with the command `sleep 5`.
3. Start the internal web server with `vovservermgr config webprovider internal`.

Restarting the Webserver

Complete the following steps to restart the webserver without bringing down vovserver.

1. Enter the following:

```
vovservermgr config webport 0
```

2. Wait five seconds, then enter:

```
vovservermgr config webport $VOV_WEB_PORT_NUMBER
```

VOV Protocol Summary

The Altair Accelerator VOV Communication Protocol is TCP-based and is a proprietary implementation that is built on a client-server model. It also uses proprietary encoding and has additional protection against snoop/replay attacks. This protocol has a default timeout of three seconds. To change the default timeout, the parameter, `VOV_RELIABLE_TIMEOUT`, can be configured.

Two parameters can affect buffers, *maxBufferSize* and *maxNotifyBufferSize* that can be configured in the `policy.tcl` file, which is located in the server working directory (SWD). Following is a summary of the acceptable values for these parameters:


Parameter	Default	Min.	Max.
maxBufferSize	16M	1K	100M
maxNotifyBufferSize	1M	100K	40M

For more information about configurations, please refer to the [Server Configuration](#) page.

The primary mechanism that utilizes the VOV protocol is the `vovsh` utility. `vovsh` implements the VTK API and connects to the vovserver. The `vovsh`, and the job wrappers `vw`, `vrt` and `vov` can also be used with the VOV protocol.

About Port Numbers

Altair Accelerator products use distinct ports. Typically, one port is the default listener for the initial vovserver. To compute this port number, the project name is normally used to compute the number of the TCP port on which the vovserver listens for connections. This is done by hashing the project name into the default port range 6200-6455, using the function `vtk_port_number`. The port range can be changed by using the environment variable `VOV_PORT_NUMBER`

 **Note:** The port range of 6200-6455 is the default range when the `VOV_PORT_NUMBER` variable is either unset or is set to `automatic`.

The following table shows examples of mapping product names to ports:

Product	Default Port	Use
Accelerator	6271	Authenticated
Accelerator	6272	Read-Only
Allocator	8787	-
Monitor	5555	Authenticated
Monitor	5556	Read-Only
FlowTracer	6200-6455	vovserver Ports

Clients can connect to the vovserver and interact using either the proprietary VOV protocol or HTTP.



Note: Additional information about the VOV protocol is proprietary and is not published. If additional information about the VOV protocol is needed, please contact [Altair Support](#).

Getting the Port Number with vovsh

If writing a script using `vovsh`, finding the port number to which the script is connected can be located by using `vtk_generic_get`. The following example assumes the Altair Accelerator Monitor is owned by "cadmgr" and is using port 5555.

```
% vovproject enable -u cadmgr licmon
% vovsh -x 'vtk_generic_get project P; puts $P(port) '
55555
```

Conflicts with the Port Number

If different names are mapped to the same port number, the corresponding servers cannot run on the same host at the same time. (This is an operating system restriction, in which only one process can listen on a host:port at a time.)

If this issue occurs, it can be resolved with one of the following actions: change host, change the project name, or explicitly choose the port number by using the environment variable `VOV_PORT_NUMBER`. This variable is defined in the `setup.tcl` file, which is located in the server configuration directory.

Time Tolerance

The file server (or servers) and the vovserver may be unsynchronized. If that occurs, comparing the timestamp of a file with the start time of a job, may produce an incorrect result due to the clock offset.

While it is easy to [synchronize the clocks](#), it may be occasionally necessary to run VOV on a network that is slightly offset.

Tolerance can be set to accept timestamp comparisons with a tolerance of a few seconds. A typical value for the tolerance is less than 10 seconds. By default, no tolerance is allowed; it is assumed that the network is synchronized.

Setting the Tolerance

Time tolerance is set by adding a line in the `policy.tcl` file. Example:

```
set config(timeTolerance) 2
```

The tolerance can also be set via the browser interface. To do so, log in as ADMIN and visit the Admin page.

Client Limitation and Tuning

The maximum number of clients - the combination of vovtaskers, user interfaces and proxies, that can be concurrently connected to a vovserver is limited by the number of file descriptors available.

This is an operating system parameter, and is inherited from the shell that starts vovserver. It can not be changed after vovserver starts.

There are two kinds of limits, a hard limit and a soft limit. Limits are imposed to reduce the likelihood of exhausting system resources. A soft limit may be set by shell commands so long as a value less than or equal to the hard limit is selected. The hard limits for descriptors and other resources may vary by user, group, and other attributes.

On UNIX, this number is operating system dependent. In most UNIX installations, the hard limit for file descriptors is 1024 or more.

On Windows NT, VOV sets the limit at 256 file descriptors.

On Linux, `root` can change the limits in the file `/etc/security/limits.conf`. Example:

```
* hard nofile 8192
* soft nofile 2048
```

The above example sets the soft limit for all users to 2048, and the hard limit to 8192. The '*' character could be replaced by that of the Accelerator owner account, e.g. 'cadmgr'.

Background

Each operating system offers a limited number of file descriptors for each process. In modern systems, this limit may be up to 65000. The vovserver can handle as many clients as the "descriptors limit" allows. It is also possible to reduce the number by setting a soft limit using the methods described above.

To allow a large number of clients, vovserver must be started with a high limit. The `ncmgr` command reports the number at startup time, please read it carefully before replying 'yes'. Example:

```
% limit descriptors 16000
% ncmgr start
```

The file descriptors are used by the vovserver to communicate with the clients: vovtaskers, GUI, browser, interactive jobs, etc.

The utilization of file descriptors are approximately:

- The server by itself needs about 10
- The other descriptors less than 40 are not used
- Each vovtasker needs 1
- Each running batch job needs 1
- Each running interactive job needs 2
- Each vovconsole needs 2
- Each nc monitor needs 1

Example: On a loaded farm with 500 vovtaskers, each with four CPUs, with half jobs interactive, the estimated number of descriptors needed is:

```
10 + 500 + 500 * 2 + 500 * 2 * 2 = 3510 file descriptors
```

This leaves descriptors for about 580 monitors and GUIs.

Behavior with Exhausted File Descriptors

The exhaustion of file descriptors rarely occurs. Altair Accelerator's main concern is preserving the integrity of the vovserver. Commands that attempt a new connection to the vovserver fail to connect and return an error message `too many clients in the system`. The vovserver will then post an alert showing `too many open files`. In that condition, ordinary commands such as `nc hosts` and `nc list` will not work because the `vovsh` that runs those commands cannot connect to vovserver.

Reserved Connection on the localhost

When file descriptors are exhausted, you can connect to vovserver using a special method through the software loopback interface (lo0, 127.0.0.1). This is achieved by setting `VOV_HOST_NAME` to `localhost`. Example:

```
% vovproject enable vncNNNN
% setenv VOV_HOST_NAME localhost
% vsi
```

Solutions to File Descriptor Exhaustion

A short-term solution is to stop some of the clients is to lower the demand for file descriptors. Transient GUI clients such as VovConsole, monitors, and Accelerator GUI should be stopped first. Any idle vovtaskers should also be stopped. Guidelines:

- Check how users are submitting jobs. There are some limits on `maxNormalClients` and `maxNotifyClients` in the `policy.tcl` file to prevent accidental or malicious denial-of-service

attacks. Sometimes we have seen jobs submitted with the `-wl` option and placed in background, each consuming 2 descriptors.

- Next you should first find whether it is possible to raise the descriptor limit on the current host. If not, a longer-term solution is to move the vovserver to another host that offers more file descriptors. A newer versions of UNIX is a good candidate as they offer 65K descriptors.

It is possible to continue operation, even in the presence of interactive jobs, by moving the vovserver and making the new queue the default queue so that newly-submitted jobs go to the new default queue. The server on the host with limited descriptors will finish all jobs, and it may then be shut down.

Client Service Modes

On Linux-based systems, there are two client servicing modes from which to choose: `poll` (default) and `epoll`. The mode chosen specifies which POSIX mechanism the vovserver will use to determine which client file descriptors are ready for use. The mode can be specified by setting `config(useepoll)` to 0 (`poll`, default) or 1 (`epoll`) in the `SWD/policy.tcl` file.

Generally, the `epoll` mode should result in more efficient processing of service requests. As of this version, `epoll` mode is a new feature and is therefore disabled by default.

Safe Server Threads

As the server farm becomes larger and larger, it becomes useful to allow the vovserver to use multiple threads to handle the costly read-only services, such as listing sets of jobs or retrieving the complete list of taskers.

This capability is controlled by the parameter `thread.service.max`, which has a default value of 2. To completely turn off this capability, set the parameter to 0. For example:

```
# In policy.tcl file:  increase number of concurrent threads allowed.
set config(thread.service.max) 8
```

Turn Off Threads

```
# In policy.tcl file:  turn off completely threads.
set config(thread.service.max) 0
```

Recommendation on Linux Systems

Safe threads are forked versions of vovserver, which should live no more than 60 seconds. Typically, these threads need to live for only a few seconds.

If the memory footprint of the vovserver is a significant fraction of the total virtual memory on the job host, the number of threads can be increased by setting the kernel parameter `vm.overcommit_memory` to 1.

For example:

```
# Example:
# sysctl -w vm.overcommit_memory=1
```

```
# Optional  
# sysctl -w vm.overcommit_ratio=50
```

Set the Range for VovId

A VovId is a nine-digit string assigned by VOV to each object in the trace.

Example: "000012345"

The VovIds grow monotonically up to 999,999,999, after which they cycle back down to restart at about 5,000.

Change the Range used for VovIds

The range used for VovIds can be configured from the command line by using the configuration keyword *idrange* and specifying both the low and the high range of the VovIds. There are rules for the relationship between the low and high values for VovId. For example, high must be at least 1,000,000 more than low. These rules are silently enforced.


In the following example, the VovId is constrained to the range of 5 million to 100 million:

```
% vovsh -x 'vtk_server_config idrange 5000000-100000000'
```

The current range of the VovIds can be retrieved with the following command:

```
% vovsh -x 'vtk_generic_get policy a; parray a vovId*  
a(vovIdHigh) = 100000000  
a(vovIdLow)  = 5000000
```

VovId Changes in FlowTracer (only in versions before 2015.09)

 **Note:** This section does not apply for version 2015.09!

In FlowTracer (but not in Accelerator), the VovId of a job changes every time the job is executed. This happens because a new node is created each time a job starts: a new node is assigned a new VovId. For the duration of the execution, there are two nodes that represent the same job:

- The node with the old VovId and with status *RETRACING
- The node with the new VovId and with status *RUNNING

Upon completion of the job, the new node remains while the old node is forgotten.

The FlowTracer server remembers the relationship between the old VovId and the new VovId; if a job node is requested with the old VovId, the server will retrieve the job with the new VovId.

VovId No Longer Changes in FlowTracer (starting from version 2015.09)

The job that is running on a tasker goes through the status RETRACING=ORANGE to RUNNING=YELLOW but does not change the VovId. The information about which dependencies are new and which ones are old is kept in the "origin" field of the dependencies.

Server Configuration Parameters

Name	Type	Default	Range	Description
<code>acl.default.jobs.</code>	string	VIEW,EXISTS,CHOW		Default ACL for FlowTracer jobs assigned to the leader role.
<code>alerts.max</code>	integer	50	[5--1000]	Maximum number of alerts. If more alerts are created, the oldest with the lowest level is archived.
<code>allowcoredump</code>	boolean	0	0,1	If nonzero, request the vovserver to dump core even if it was started from a shell where limitcoredumpsize was zero.
<code>allowForeignJobsO</code>	boolean	0	0/1	Allow jobs from other users to run on non-root taskers. Default=0, set to 1 to allow scheduling of jobs on non-root taskers even if the job belongs to a different user than that of the tasker.
<code>allowUidForSecuri</code>	integer	0	[0--2147483647]	Additional UID that can own the security.tcl file
<code>alm.enable</code>	boolean	0	0/1	Enable Altair Licensing.
<code>autoForgetFailed</code>	time spec	2 days	[5m,infinity]	If the autoForget bit is set on a job, the job will

Name	Type	Default	Range	Description
				be forgotten automatically upon failure after the specified time.
<i>autoForgetOthers</i>	time spec	2 days	[5m,infinity]	If the autoForget bit is set on a job, the job will be forgotten automatically if it is not scheduled after the specified time.
<i>autoForgetRemoveL</i>	boolean	0	0/1	Try to remove the log files associated with auto-forgotten jobs.
<i>autoForgetValid</i>	time spec	1 hour	[5m,infinity]	If the autoForget bit is set on a job, the job will be forgotten automatically upon successful termination after the specified time.
<i>autoLogout</i>	time spec	4 hours	[10m,infinity]	The period of inactivity after which the system automatically logs a user out of the browser interface. The minimum value for this parameter is 10 minutes.
<i>autoRescheduleCou</i>	integer	4	0-10	Specifies the maximum number of reschedule attempts per job.
<i>autoRescheduleOnN</i>	boolean	1	0/1	Controls whether auto-rescheduling will avoid the same host (default) or

Name	Type	Default	Range	Description
				just the tasker, allowing the job to run on a different tasker on the same host. Does not apply to auto-rescheduling based on exit status (see exit-status docs).
<code>autoRescheduleThr</code>	time spec	0s	0s-60s	This parameter deals with jobs that fail quickly. If the failure time is less or equal to the specified value, the job is automatically resubmitted at high priority, with the name of the failing host as a negated resource, so that the job will be placed on a different host (or tasker, configurable) for the next run. If the <code>autoRescheduleThreshold</code> is 0, automatic resubmission is disabled. A maximum of 4 auto-reschedules are permitted by default. This maximum can be controlled via the <code>autoRescheduleCount</code> parameter. Host/tasker behavior can be controlled via the <code>autoRescheduleOnNewHost</code> parameter. Does

Name	Type	Default	Range	Description
				not apply to auto-rescheduling based on exit status (see exit-status docs).
<code>autostop.timeout</code>		60s		<p>This parameter provides a timeout for the execution of all scripts installed in the SWD/autostop directory. The autostop function is called automatically upon shutdown of the vovserver and all scripts have up to the timeout to executed and exit. This time period is provided to allow autostop scripts to communicate with the vovserver before exiting, while protecting against a problematic script that fails to exit. Once the timeout has been reached, the vovserver continues with its exit procedure, and any autostop scripts that have opened a connection to the vovserver will likely fail. Default can be specified in <code>policy.tcl</code>.</p> <p>Example: <code>set config(autostop.timeout) 120</code></p>

Name	Type	Default	Range	Description
				This parameter can also be set using the VTK API. Example: <pre>vtk_server_config autostop.timeout 120</pre>
<i>blackholeDiscardT</i>	timespec	10m00s	[2m00s--10h00m]	Failed jobs older than this time are discarded in the blackhole computation
<i>blackholeFailRate</i>	double	0.900000	[0.250000--1.00000]	Fraction of failed jobs that are running well on another tasker
<i>blackholeFailedJo</i>	integer	5	[2--10000]	Number of failed jobs on a tasker to consider the tasker a black hole
<i>blackholeMaybeTim</i>	timespec	10s	[0s--1h00m]	Suspension time for tasker that maybe a black hole, but we are not sure yet
<i>blackholeSuspendT</i>	timespec	3m00s	[1m00s--10h00m]	Suspension time for a black hole
<i>bucketValidTasker</i>	timespec	5m00s	[0s--41d16h]	Control frequency of determining if any valid taskers exist for a bucket. If this parameter is 0, then the facility is turned off.
<i>cgi.max</i>	integer	40	[10--200]	Max number of concurrent CGI pages that can be generated.

Name	Type	Default	Range	Description
<code>checkoutHostLower</code>	boolean	0	0/1	If set, host name is always forced to lower case in checkouts.
<code>checkoutUserLower</code>	boolean	0	0/1	If set, user name is always forced to lower case in checkouts.
<code>clientConnectionQ</code> <code>clientConnectionQ</code>				By default, the vovserver responds to multiple incoming connections during the same cycle. This feature can be controlled with these two parameters: <code>clientConnectionQueue.mode</code> multi or single, default is multi. <code>clientConnectionQueue.size</code> the size, integer value 25-1024, default is 512.
<code>comm.buffer.compr</code>	integer	1	[1--9]	Compression level between 1 and 9. 1 is speed first and 9 is percentage of string data in buffer to trigger buffer compression.
<code>comm.buffer.compr</code>	integer	4000	[100--1000000]	Minimum packet size to trigger buffer compression.
<code>comm.buffer.compr</code>	integer	50	[1--100]	Minimum percentage of string data in buffer to

Name	Type	Default	Range	Description
				trigger buffer compression.
<code>comm.maxBufSize</code>	unsigned	16777216	[65536--214748364]	Maximum size of communication buffers between server and client.
<code>comm.maxTimeToCom</code>	integer	5000	[1000--20000]	
<code>comm.minBufSize</code>	unsigned	32	[32--2147483647]	Minimum size of communication buffers between server and client.
<code>comm.minTimeForCo</code>	integer	1000	[1000--2000]	
<code>comm.mmapThreshol</code>	unsigned	4294967295	[4096--4294967295]	Size at which comm buffers are allocated via mmap.
<code>comm.requireClie</code>	boolean	0	0/1	Require clients authenticate with a vov security key.
<code>comm.tls.enable</code>	boolean	1	0/1	Enable TLS for vov protocol clients.
<code>cpuprogressWindow</code>	integer	1	[1--1440]	Number of samples to average when calculating cpuprogress (allowed range: 1-1440, default value: 1)
<code>crashRecoveryMaxE</code>	timespec	1m00s	[0s--30m00s]	Max extension of crash recovery period due to taskers reconnecting during quiet time.
<code>crashRecoveryPeri</code>	timespec	60s	[30s,5m]	This parameter specifies how long the server is in crash recovery

Name	Type	Default	Range	Description
				mode. The default value is for 60 seconds after which the server will enter normal operation.
<i>crashRecoveryQuiet</i>	timespec	30s	[0s--5m00s]	Possible crash recovery delay due to tasker reconnection during quiet time before crash recovery ends.
<i>dashboard.threshold</i>	integer	300	[1--1000000000]	Minimum number of buckets to mark as "critical" bucket count in the web dashboard. (default: 300)
<i>dashboard.threshold</i>	integer	200	[1--1000000000]	Minimum number of buckets to mark as "high" bucket count in the web dashboard. (default: 200)
<i>dashboard.threshold</i>	integer	50	[1--1000000000]	Minimum number of buckets to mark as "low" bucket count in the web dashboard. (default: 100)
<i>dashboard.threshold</i>	integer	100	[1--1000000000]	Minimum number of buckets to mark as "medium" bucket count in the web dashboard. (default: 100)
<i>dashboard.threshold</i>	double	3.000000	[0.001000--30.0000]	Minimum scheduler time to mark as "critical" scheduler activity in the web

Name	Type	Default	Range	Description
				dashboard, in seconds. (default: 3.0)
<i>dashboard.thresho</i>	double	1.000000	[0.001000--30.0000	Minimum scheduler time to mark as "high" scheduler activity in the web dashboard, in seconds. (default: 1.0)
<i>dashboard.thresho</i>	double	0.100000	[0.001000--30.0000	Minimum scheduler time to mark as "low" scheduler activity in the web dashboard, in seconds. (default: 0.5)
<i>dashboard.thresho</i>	double	0.500000	[0.001000--30.0000	Minimum scheduler time to mark as "medium" scheduler activity in the web dashboard, in seconds. (default: 0.5)
<i>dashboard.thresho</i>	integer	90	[0--100]	Minimum percentage to mark as "critical" server size in the web dashboard, as a percentage of total host RAM. (default: 90)
<i>dashboard.thresho</i>	integer	60	[0--100]	Minimum percentage to mark as "high" server size in the web dashboard, as a percentage of total host RAM. (default: 60)

Name	Type	Default	Range	Description
<code>dashboard.thresho</code>	integer	20	[0--100]	Minimum percentage to mark as "low" server size in the web dashboard, as a percentage of total host RAM. (default: 40)
<code>dashboard.thresho</code>	integer	40	[0--100]	Minimum percentage to mark as "medium" server size in the web dashboard, as a percentage of total host RAM. (default: 40)
<code>defaultStopSignal</code>	string	TERM,HUP,INT,KILL		Default stop signal cascade; used for job control.
<code>defaultStopSignal</code>	integer	1	[0--20]	Default stop signal delay; number of seconds to wait between transmitting signals to taskers when using vovstop and related commands. (allowed range: 0-20, default: 1)
<code>defaultSuspendSig</code>	string	STOP		Default suspend signal; used for job control
<code>disablefileaccess</code>	boolean	0		If this parameter is greater than 0 (zero), the server will not access any file that is not under \$VOVDIR or \$VNCSWD. By 'access' we mean

Name	Type	Default	Range	Description
				<p>any operation on the file like stat(), open(), access(). This is a protection to prevent a possible server freeze if it tries to access files on an dead NFS server. If this parameter is greater than 1 (one), then not only the server disables access to files, but also the CGI clients that could possibly show file information will disable such functionality. In summary:</p> <ul style="list-style-type: none"> • Under normal conditions, set this parameter to 0 • If you are concerned about the vovserver blocking on a bad NFS mount, set this parameter to 1 • If you want maximum information security, set this parameter to 2.
<i>diskspacecheck.mi</i>	unsigned	1000	[1--50000]	Specify free disk space alert

Name	Type	Default	Range	Description
				generation threshold (in MB) for free disk space checking (allowed range: 1-50000, default: 1000)
<i>diskspacecheck.mi</i>	integer	1	[1--50]	Specify free disk space alert generation threshold (in %) for free disk space checking (allowed range: 1-50, default: 1)
<i>emptyBucketAge</i>	integer	120	[0--86400]	Delete empty buckets older than the specified age in seconds.
<i>enableBackfillRes</i>	boolean	0	0/1	Enable/Disable backfilling jobs on resource reservation.
<i>enableLdap</i>	boolean	0	0/1	If set, the system uses also LDAP for browser authentication. Available on the following architectures: macosx, linux64, armv7l
<i>enableLookAheadRe</i>	boolean	0	0/1	Enable/Disable job based reservation.
<i>enablePam</i>	boolean	1	0/1	If set, the system uses PAM for browser authentication.

Name	Type	Default	Range	Description
<i>enablePartialRese</i>	boolean	0	0/1	Enable/Disable partial tasker reservation.
<i>enableTimestampCh</i>	boolean	1	0/1	On large traces, it is good to completely disable the checking of the timestamps of files.
<i>enableTimestampCh</i>	boolean	1	0/1	On large traces, it is good to disable the checking of the timestamps of the upcone of a target because it may take too long.
<i>enableUserArtifac</i>	boolean	0	0/1	Enable/Disable cleanup of user-created artifacts including fairshare groups, sets, and limit resources that are no longer in use.
<i>enableWaitReasons</i>	boolean	1	0/1	Enable logging of Wait-Reasons.
<i>enterpriselicense</i>	string	Auto		Enterprise licensing mode
<i>enterpriselicense</i>	boolean	0	0/1	Use burst licenses if available.
<i>enterpriselicense</i>	timespec	5m00s	[10s--1h00m]	With enterprise licensing in automatic mode, wait this much before decreasing again.
<i>enterpriselicense</i>	timespec	4s	[1s--5m00s]	With enterprise licensing in automatic mode,

Name	Type	Default	Range	Description
				wait this much before increasing again.
<i>epollbuf</i>	integer	50	[50--1024]	Epoll buffer size
<i>failover.maxdelay</i>	integer	120	[10--3600]	Specifies the maximum duration during which a vovtasker can participate in a failover server election (default: 120).
<i>failover.usefailo</i>	boolean	0	0/1	If set, only taskers in the failover tasker group participate in the failover server election (default: 0).
<i>fairshare.allowAd</i>	boolean	0	0/1	Allow ADMIN to bypass ACL when modifying a FairShare group.
<i>fairshare.compute</i>	boolean	1	0/1	Enable/disable FairShare computation
<i>fairshare.default</i>	integer	100	[0--100000]	Default weight assigned to new FairShare groups. Can also be controlled by a sibling group called 'default'.
<i>fairshare.default</i>	timespec	1h00m	[0s--7d00h]	Default window assigned to new FairShare groups. Normally the groups inherit the window from their parent of from a

Name	Type	Default	Range	Description
				sibling group called 'default'.
<i>fairshare.maxjobs</i>	integer	20	[1--1000000]	Max number of jobs to dispatch from each bucket at once.
<i>fairshare.maxjobs</i>	integer	20	[1--1000000]	Max number of jobs to dispatch on each loop, after which the FairShare stats will be recomputed.
<i>fairshare.nocompu</i>	timespec	2m00s	[1m00s--1h00m]	If FairShare computation is disabled and FairShare has not been updated externally, maximum time to wait (in seconds) before computing FairShare.
<i>fairshare.oversho</i>	boolean	1	0/1	If set, each bucket will be limited by its max dispatch depth for each scheduler cycle (default: 1).
<i>fairshare.relativ</i>	integer	1	[0--2]	Assuming: t=target allocation; h=historic allocation in window; r=running allocation now. If set to 2, compute FairShare as a weighted sum of $d = (r-t) + \alpha(h-t)$. If set to 1, compute FairShare distance relative

Name	Type	Default	Range	Description
				to the FairShare target $d=((r-t)+(h-t))/t$ (of course assuming $t>0$). If set to 0, compute FairShare distance by a simple difference between actual and target $d=(r-t)+(h-t)$.
<i>fairshare.relative</i>	double	10.000000	[0.001000--1000.00]	Weight of historic distance relative to running distance in computing value for FairShare ranking. Only used if fairshare.relative is set to 2.
<i>fairshare.strictN</i>	integer	0	[0--1]	Perform strict acl checks when creating non-USER type nodes. If set to 0, user only needs ATTACH acl on parent. If set to 1, user needs CREATE acl on parent.
<i>fairshare.updateP</i>	timespec	4s	[0s--1h00m]	Update FairShare stats no more frequently than specified period.
<i>hero.to.monitor.f</i>	integer	30	[1--10000]	How often to send resource usage updates from Hero to Monitor.
<i>hero.to.monitor.u</i>	boolean	0	0/1	Enable resource usage updates from Hero to Monitor

Name	Type	Default	Range	Description
<i>hog.protection.cl</i>	integer	1	[1--600]	Specify the client processing delay (in seconds) for users under hog protection (allowed range: 1-600, default: 1).
<i>hog.protection.en</i>	boolean	0	0/1	Enable hog protection in scheduler; note that this is independent of <i>hog.protection.time.enable</i> (default: 0)
<i>hog.protection.job</i>	integer	100000	[1000--9999999]	Specify currently-managed (all states) job count threshold to trigger hog protection against a user (allowed range: 1000-9999999, default: 100000)
<i>hog.protection.time</i>	double	1.000000	[1.000000--100.000]	Multiple of allocated service time required before classifying a user as a hog (allowed range: 1.0-100.0, default: 1.0)
<i>hog.protection.time</i>	double	1.000000	[0.100000--100.000]	Multiple of service-time exceeding allocated service time to apply as a delay for service hogs (allowed range: 0.1-100.0, default: 1.0)
<i>hog.protection.time</i>	boolean	1	0/1	Enable time-based hog protection

Name	Type	Default	Range	Description
				in scheduler; note that this is independent of hog.protection.enable (default: 1)
<i>hog.protection.time</i>	integer	30	[1--300]	Lookback window when computing time-based service hogs, in seconds (allowed range: 1-300, default: 30)
<i>hog.protection.time</i>	double	2.000000	[0.000000--100.000]	Minimum percentage of real-time taken by a user's services to be considered an "active user" for service hog calculations (allowed range: 0.0-100.0, default: 2.0)
<i>hog.protection.time</i>	double	40.000000	[0.000000--100.000]	Minimum percentage of real-time taken by all services to enable time-based service hog calculation (allowed range: 0.0-100.0, default: 40.0)
<i>http.fileAccessDefault</i>	string	ALLOW-OWNER		Whether to ALLOW, ALLOW-OWNER, or DENY file access as the default for REST based file access calls.
<i>http.maxReadOnlySession</i>	string	30d		The maximum time a read-only session can be valid before it expires.

Name	Type	Default	Range	Description
<i>http.maxSSLVersion</i>	string	TLSv1.3		The maximum SSL/TLS version supported by the internal webserver.
<i>http.minSSLVersion</i>	string	TLSv1.3		The minimum SSL/TLS version supported by the internal webserver.
<i>http.proxytimeout</i>	unsigned	60	[5--10000]	HTTP Proxy Timeout (seconds)
<i>httpSecure</i>	boolean	1		This should be set to 1 or "enable" if you use the browser user interface. If it is set to "0", browser users do not have to authenticate, and hence the server does not know their login identity. The default value is 1. When this parameter is 1, the server requires basic authentication to serve most of its pages. By setting this parameter to 0 (zero), any access via the HTTP protocol is allowed, but the server will not know user's login identity. This is provided for backwards compatibility, but the use of the value is discouraged. If

Name	Type	Default	Range	Description
				this parameter is set to 2, or "disable", or any other number, then the HTTP interface is completely disabled.
<i>http.tls12Ciphers</i>	string	AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:AES128-GCM-SHA256		The list of ciphers supported by TLS 1.2 and older when using the internal webserver.
<i>http.tls13Ciphers</i>	string			The list of cipher suites supported by TLS 1.3 when using the internal webserver.
<i>http.workerthread</i>	unsigned	5	[2--50]	HTTP Worker Threads. Change requires vovserver restart.
<i>isProductionSetup</i>	boolean	0	0/1	Specifies whether this queue is deployed in a production environment.
<i>jetstreams.enable</i>	boolean	1	0/1	Enable jetstreams in scheduler (default: 1)
<i>jetstreams.jobLim</i>	integer	100	[0--2147483647]	Limit the number of jobs dispatched per jet stream (allowed range: 0-INT32_MAX, default: 100, 0 = unlimited)
<i>jetstreams.thresh</i>	integer	5	[1--86400]	Specify maximum job duration (in

Name	Type	Default	Range	Description
				seconds) for utilizing jetstreams (allowed range: 1-86400, default: 5)
<code>jobprofile.usedb</code>	boolean	0	0/1	If set to 0, save job profile to in-memory wave. If set to 1, save job profile to time-series database.
<code>keptJobsCleanupCh</code>	integer	10000	[100--1000000]	keepfor jobs cleanup chunk size.
<code>keptJobsCleanupIn</code>	timespec	5m00s	[1m00s--1h00m]	Interval in seconds for kept jobs cleanup.
<code>la.adjustForUncon</code>	boolean	1	0/1	Enable/disable adjust for unconsumed allocations.
<code>la.adjustForUncon</code>	timespec	2m00s	[1m00s--5m00s]	Time window to check for unconsumed allocations for adjustment.
<code>la.feature.expira</code>	timespec	1h00m	[30s--1d00h]	Time threshold since feature data last received from LM, after which LA will stop allocating this expired feature.
<code>la.ooq.waves.disa</code>	boolean	0	0/1	Disable wave based OOQ estimation.
<code>license.AltairUni</code>	boolean	0	0/1	Set to 1 to enable Altair Units licensing.

Name	Type	Default	Range	Description
<i>license.promise</i>		7200	1 minute more than the refresh time, but greater than 20 - 7200.	Specifies the time, in minutes, past the most recent refresh when the RLM license will be checked in. Default: 7200.
<i>license.refresh</i>		60m	10m-1 minute less than the promise time.	Specifies the rate, in minutes, for the vovserver to refresh its RLM license checkout.
<i>liverecorder.logd</i>	string	/tmp		Directory in which the Live Recorder recording file should be saved. The directory must exist. Default is '/tmp'.
<i>liverecorder.logs</i>	integer	256	[256--65536]	Live Recorder log size in MB (range: 256-65536 default: 256)
<i>lm.lookAndFeel</i>	string	latest		License Monitor 'look and feel' version. Default is 'latest'.
<i>log.nodestatus.ch</i>	boolean	0	0/1	Enable additional detail on node status changes.
<i>log.nodestatus.lo</i>	integer	0	[0--3]	Global nodestatus logging type (0=NONE, 1=ALL, 2=INVALID, 3=DOWNCONE)
<i>logs</i>	boolean	0	0/1	Enable tasker log compression.
<i>logs,checkouts,co</i>	boolean	1		You can control which log files

Name	Type	Default	Range	Description
				get compressed by editing the policy.tcl file and setting the config(log,XXXX,compress) flags.
logs,checkouts,handler	string			A command line to process the log file as it is being created. The command line should contain the variables @FILENAME@ and @EXPIRE@.
logs,jobs,compress	boolean			You can control which log files get compressed by editing the policy.tcl file and setting the config(log,XXXX,compress) flags.
logs,jobs,handler	string			A command line to process the log file as it is being created. The command line should contain the variables @FILENAME@ and @EXPIRE@.
logs,journals,compress	boolean	0		You can control which log files get compressed by editing the policy.tcl file and setting the config(log,XXXX,compress) flags.

Name	Type	Default	Range	Description
<code>logs,journals,handle</code>	string			A command line to process the log file as it is being created. The command line should contain the variables @FILENAME@ and @EXPIRE@.
<code>logs,resources,compress</code>	boolean	0		You can control which log files get compressed by editing the <code>policy.tcl</code> file and setting the <code>config(log,XXXX,compress</code> flags.
<code>logs,resources,handle</code>	string			A command line to process the log file as it is being created. The command line should contain the variables @FILENAME@ and @EXPIRE@.
<code>logs,server,compress</code>	boolean	0		You can control which log files get compressed by editing the <code>policy.tcl</code> file and setting the <code>config(log,XXXX,compress</code> flags.
<code>logs,server,handle</code>	string			A command line to process the log file as it is being created. The command line should contain the variables

Name	Type	Default	Range	Description
				@FILENAME@ and @EXPIRE@.
<i>logs,tasker,compress</i>	boolean	0	0/1	You can control which log files get compressed by editing the policy.tcl file and setting the <code>config(log,XXXX,compress flags</code> .
<i>logs,waitreason,c</i>	boolean	0		You can control which log files get compressed by editing the policy.tcl file and setting the <code>config(log,XXXX,compress flags</code> .
<i>logs,waitreason,h</i>	string			A command line to process the log file as it is being created. The command line should contain the variables @FILENAME@ and @EXPIRE@.
<i>longServiceWarnin</i>	double	0.800000	[0.000000--1000000]	Warn if a service takes a long time, as defined by this parameter (in seconds).
<i>lookAheadSchedInt</i>	timespec	1s	[0s--1m00s]	Interval in seconds for lookahead scheduling.
<i>lookAheadTimeReso</i>	timespec	1m00s	[1s--1h00m]	Interval of discrete time instances at which available resource count

Name	Type	Default	Range	Description
				is maintained, in seconds.
<i>maxAgeRecentJobs</i>	timespec	60	[0,3600]	This defines the max age of the jobs in the set System:transitionsRecent, which contains all the jobs that have recently finished. Due to various scheduling delays, it is possible for some jobs in the set to be somewhat older.
<i>maxBufferSize</i>	unsigned	16777216	[65536--214748364]	Maximum size of communication buffers between server and client. The same with comm.maxBufSize.
<i>maxChildren</i>	integer	20		Maximum number of children managed by vovserver. This mostly affects the number of concurrent CGI requests that the server can manage.
<i>maxCommandLineLen</i>	integer	40960	[128,100000]	Number of characters in the longest command line that will be accepted by the system. A job containing a longer command line will be rejected.

Name	Type	Default	Range	Description
<i>maxEnvLength</i>	integer	512	[128,16000]	Maximum length allowed for the environment field in a job. Example: If the environment is BASE+GNU, length of the string is 8.
<i>maxJobArray</i>	integer	10000	[10,100000]	This parameter defines the maximum size of a job array.
<i>maxLevel</i>	integer	4095	[3,4095]	Maximum number of levels in the dependency graph. Dependency declarations that cause the levels in the graph to exceed this value are rejected.
<i>maxLookAheadReser</i>	integer	10	[1--1000]	Maximum lookahead reservation count at a given time.
<i>maxLookAheadSched</i>	timespec	4h00m	[1h00m--1d00h]	Maximum lookahead reservation scheduling timeout in hour.
<i>maxNormalClients</i>	integer	400		Maximum number of normal clients per user. This is a protection against denial-of-service attacks by a single user.
<i>maxNotifyBufferSi</i>	integer	1000000	[100k,40M]	In bytes, this parameter controls the size of the

Name	Type	Default	Range	Description
				buffers used internally to notify asynchronous clients. If you see "overflow" events and are annoyed by it, you can increase the size of the buffer.
<i>maxNotifyClients</i>	integer	40		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.
<i>maxORsInResourceE</i>	integer	20	[0--25]	Max number of OR operators in a resource expression.
<i>maxPathLength</i>	integer	1024	[128,16000]	Maximum length of names of files, including the end of line terminator. This is different from all other "max" parameters for jobs. This means that if this parameter has value 1024, then the actual max length of a path is actually $1024-1=1023$.

Name	Type	Default	Range	Description
<i>maxResMap</i>	integer	5000	[1, infinity]	Maximum resources map number that can be set in the resource map set.
<i>maxResourcesLength</i>	integer	1024	[128,16000]	Maximum length of the resources field for a job.
<i>maxSmartSets</i>	integer	50	[10--5000]	Max number of smart sets.
<i>maxeventqueue</i>	integer	100000000	[1000--500000000]	Max number of events in event queue.
<i>metrics.autoSavePeriod</i>	timespec	5m00s	[1m00s--1h00m]	Period for auto-saving metrics data files to disk.
<i>metrics.enable</i>	integer	7	[0--7]	Enable the metrics system. The old name 'enableMetrics' can also be used. The value is a mask, with the following bit values: 0=disable_all_metrics 1=enable_server_metrics 2=enable_design_metrics 4=enable_fairshare_metrics 7=enable_all_metrics
<i>metrics.keepFiles</i>	timespec	30d00h	[0s--3y00d]	How long to keep saved metrics data files.
<i>metrics.logdir</i>	string	/tmp		Directory in which the metrics file should be saved. The directory must exist. Default is tmp.

Name	Type	Default	Range	Description
<i>metrics.maxmem</i>	integer	2000	[100--100000]	Max size in MB for each metrics family. If the memory exceeds this amount, the system drops some metrics or some points.
<i>metrics.usedb</i>	boolean	0	0/1	If set to 0, save metrics to in-memory metrics database. If set to 1, save metrics to time-series database.
<i>minMemFreeOnServe</i>	integer	10	[0--10000]	Issue alert if free memory on server goes below this minimum threshold, in MB.
<i>minSwapFreeOnServ</i>	integer	10	[0--10000]	Issue alert if free swap on server goes below this minimum threshold, in MB.
<i>minhw</i>	string	"RAM/20 CORES/1 SLOTS/1 PERCENT/1"		The minimum resources requested by a job.
<i>mm.fork</i>	boolean	0	0/1	Enable use of MADV_DONTFORK for client buffers.
<i>mm.huge</i>	boolean	0	0/1	Enable Huge Pages
<i>nfsdelay</i>	timespec	0s	[0s--5m00s]	Do not even check files that have changed less than 'nfsdelay' ago.

Name	Type	Default	Range	Description
<i>notifyMaxEffort</i>	integer	20	[5--50]	Percent of time to give to notification of clients
<i>notifySkip</i>	integer	0	[0--100000]	OBSOLETE: NO LONGER USED In inner loop, do not notify clients for each event, but skip a few.
<i>prSaveTimeout</i>	timespec	1h00m	[1m00s--41d16h]	Time (seconds) after which an incomplete PR save operation exits. Default is 1 hour
<i>preemption.load</i>	boolean	1	0/1	Load/unload preemption module
<i>preemption.log.all</i>	boolean	0	0/1	Enable debug messages for all preempt rules
<i>preemption.log.verbose</i>	integer	0	[0--10]	Control verbosity of log messages for preemption.
<i>preemption.max.time.rule</i>	double	0.300000	[0.001000--10.0000]	Max time spent in preemption for each loop
<i>preemption.max.time.rule</i>	double	0.300000	[0.001000--10.0000]	Max time spent in preemption for each rule. Any rule that exceeds this will be disabled.
<i>preemption.module</i>	string	libpreemption.so		Preemption module name
<i>preemption.rule.disabled</i>	time spec	"1y"	[0,"1y"]	Specifies how long rules are disabled due to exceeding preemption.max.time.rule. 1y = Indefinite, 0

Name	Type	Default	Range	Description
				= generate alert only. Default is 1y (Indefinite).
<i>preemptionFairshare</i>	timestamp	0		If greater than the current date, enable computation of statistics for FairShare of preemption. Normally set by vovpreemptd. For testing, use vtk_server_config "preemptionFairshare" "on", which activates the computation for the next 10 minutes.
<i>preemptionPeriod</i>	time spec	3s	[0,1000h]	This parameter specifies how often Accelerator will attempt to perform preemption. The default value is 3 seconds which will cause preemption to be attempted every 3 seconds. If a value of 0 is specified then this will disable preemption.
<i>preemptionResource</i>	time spec	90s	[0,1000h]	This parameter controls the frequency of releasing features for suspended jobs as a result of preemption. If this parameter is 0,

Name	Type	Default	Range	Description
				the code is never called.
<i>processRetentionT</i>	timespec	5s	[0s--5m00s]	How long a process record (see procinfo feature) is kept in memory since the last update.
<i>prop.maxNameSize</i>	unsigned	255	[16--1024]	Max length of name a property
<i>prop.maxStringSiz</i>	unsigned	131071	[128--1048576]	Max length of value of a string property. Any value longer than this will be truncated.
<i>query.processor.s</i>	boolean	0	0/1	If set to 1, use the stream processor (faster) for vovselect/ vtk_select_loop queries, when feasible and supported by the client
<i>rds.enable</i>	boolean	0	0/1	Enable LM-to-NC license resource updates over sockets instead of via vovresourced. Default: false
<i>rds.secure</i>	boolean	0	0/1	Enable security on the RDS event port. Default: false
<i>rdsMaxEffort</i>	integer	20	[5--80]	Percent of time limit for updates from RDS
<i>readonlyport</i>	integer	0	[0--65535]	The readonly port for vovserver

Name	Type	Default	Range	Description
<i>rejectIndirectTask</i>	boolean	0	0/1	Reject jobs submitted through the binary indirect tasker
<i>resMapGroupDefault</i>	integer	50	-1 (unlimited)	This parameter sets the default value for the resource map "Group:groupname" for a new group. This resource map limits the maximum number of jobs that the group can run at the same time. For each individual group, you can control the value of the map with a <code>vtk_resourcemap_set Group:groupname number</code> statement in the <code>resources.tcl</code> file. A negative value of this parameter is interpreted as "unlimited."
<i>resMapToolDefault</i>	integer		-1 (unlimited)	This parameter sets the default value for the resource map "Tool:toolname" for a new tool. This resource map limits concurrent number of jobs using the tool. For each individual tool, you can control the value

Name	Type	Default	Range	Description
				of the map with a vtk_resourcemap_set Tool:toolname number statement in the resources.tcl file. A negative value of this parameter is interpreted as "unlimited."
<i>resmap.max.map.le</i>	integer	1024	[512--64000]	Max length of the map field in a resourcemap
<i>resmap.max.maps.i</i>	integer	300	10, 50000	Max number of resourcemaps considered in the evaluation of a resource expression. This is a protection against cycles in the resource maps (e.g. a b c). The typical value is less or equal to the total number of resource maps in the system.
<i>resmap.max.nru.ti</i>	double	0.100000	[0.000100--1.00000]	Max time allowed to compute NRU handles
<i>resmap.sw.types</i>	string	License Limit Policy User Group Priority		Comma or Space separated list of types of resources that are guaranteed to be resource maps (i.e. SW resources), even if they are not currently defined

Name	Type	Default	Range	Description
				as a resource map. This parameter is typically an empty list in Accelerator Plus and includes 'License Limit Policy ...' for the other products.
<i>resources.usedb</i>	boolean	0	0/1	If set to 0, use resources data from *.SWD/data to generate wave files If set to 1, use resources data from time-series database to generate wave files
<i>restApiMaxFileSiz</i>	integer	1000	[10--10000]	REST Api max file size in KB.
<i>resuserDisableMat</i>	integer	1000	[0--1000000]	Threshold of LM handles and FT jobs to match. If the sum of LM handles and FT jobs exceeds this threshold, do not perform matching.
<i>resuserEnableNRUM</i>	boolean	1	0/1	Matching check should include checking for Not-Requested-Used licenses.
<i>resusermatchtoler</i>	timespec	0s	[0s--10m00s]	Tolerance (in seconds) allowed when matching jobs and handles
<i>resusermaxmatches</i>	integer	6	[0--40]	Maximum number of 'also' matches between jobs and lm handles

Name	Type	Default	Range	Description
<i>runAsWX</i>	integer	0	[0--1]	Set up this project to run as Accelerator Plus.
<i>sanity,smartsets</i>	boolean	0	0/1	Enable sanity check of smart sets
<i>sched.busy.thresh</i>	integer	3	[2--1000000]	If the delay between poll() is more than this threshold, then the server is considered to be 'busy'
<i>sched.maxpostpone</i>	integer	10000	[1--1000000]	Exit dispatch loop after this number of jobs have been postponed, indicating a loaded or busy farm
<i>sched.megapoll.th</i>	integer	100	[5--10000000]	Threshold that defines how many clients are needed to declare that we are in a 'mega-poll' situation
<i>sched.taskerstats</i>	boolean	0	0/1	Enable scheduler statistics for taskers (used in unusedhw.cgi)
<i>schedMaxEffort</i>	integer	20	5 50	If the scheduler becomes expensive, i.e. if it takes more than 0.02 seconds to dispatch jobs, the vovserver adaptively skips some scheduling cycle to give time to the clients to interact with the

Name	Type	Default	Range	Description
				vovserver. In this way, the scheduler effectively changes from a pure event-driven behavior to a cycle-based behavior. The cycle is controlled by the schedMaxEffort variable, which sets the ratio between the effort spent in the scheduler and the effort spend elsewhere. Example: If the schedMaxEffort is 20 (meaning 20% of time should be spent in the scheduler) and the scheduler time is 0.1s, then the scheduler will be called no more frequently than once every 0.5 seconds ($0.1 \times 100 / 20 = 0.5$)
<code>sds.enable</code>	boolean	0	0/1	Enable/disable Streaming Data Service
<code>seatlic_max</code>	integer	2147483647	[0--2147483647]	Override the maximum number of seat licenses that can be checked out (default: unlimited).
<code>setQueueEnv</code>	boolean	-	0/1	Whether to set NC_QUEUE in the job env to

Name	Type	Default	Range	Description
				Accelerator Plus project.
<i>skipConflictingJobs</i>	boolean	0	0/1	Enable/Disable processing of subsequent jobs when the top job in the bucket is conflicting with the future tasker reservation
<i>ssl.enable</i>	boolean	1	0/1	Enable SSL for web server
<i>ssl.enableRawUrl</i>	boolean	1	0/1	Enable /raw url's for SSL
<i>substituteArrayId</i>	boolean	0	0/1	Substitute references to @ARRAYID@ in resource string with array ID (default: 0). Caution should be used because this will result in a job queue bucket being created for each submission of a job array.
<i>substituteArrayIndex</i>	integer	0	[0--2]	Substitute references to @INDEX@ in resource string with array index (default: 0). Caution should be used because this will result in a job queue bucket being created for each job in an array. If set to value of 2

Name	Type	Default	Range	Description
				then find and reuse existing buckets.
<i>substituteJobIdIn</i>	boolean	0	0/1	Substitute references to @JOBID@ and @IDINT@ in resource string with job ID (default: 0). Caution should be used because this will result in a job queue bucket being created for each job.
<i>switchToExtBufThr</i>	integer	100000	[10k,1...31]	In bytes, this parameter controls the size of the internal buffer used for vtk_set_list_elements_enhar If the size exceeds this parameter, then vovserver switches to an external buffer.
<i>taskIdleReqTime</i>	timespec	2m00s	[0s--7d00h]	Required task idle time to run the user-created artifacts cleanup.
<i>tasker.attachDeta</i>	integer	-1	[-1--1]	Reserved for internal use only.
<i>tasker.authorizat</i>	integer	5	[0--60]	Minimum time in seconds that vovserver waits before authorizing new taskers
<i>tasker.childProce</i>	boolean	0	0/1	If TRUE, taskers should kill all of a job's child processes when

Name	Type	Default	Range	Description
				that job exits (i.e. assume child processes running after jobs exit are zombie processes); default is FALSE.
<code>tasker.childProcessCleanup</code>	string			If set in conjunction with <code>tasker.childProcessCleanup</code> , then taskers should kill all of a job's child processes when that job exits, except for those named here in a comma separated list. default is ""
<code>tasker.max.reservations</code>	integer	10	[0--100]	Maximum number of reservations per tasker
<code>tasker.maxWaitToReconnect</code>	timespec	4d00h	[5s--1y00d]	Time to wait after losing connection to server before reconnecting again.
<code>tasker.minWaitToReconnect</code>	timespec	5s	[1s--1y00d]	Min time to wait after losing connection to server before reconnecting again.
<code>tasker.props.enabled</code>	boolean	1	0/1	Enable saving of message properties for all taskers
<code>tasker.slotLimit</code>	integer	1000	[1--2147483647]	Maximum number of slots per tasker
<code>tasker.uninterruptible</code>	boolean	0	0/1	If TRUE, taskers will ignore incoming job

Name	Type	Default	Range	Description
				control requests if an existing request is in-process; default is FALSE.
<i>taskerBusyUponDis</i>	boolean	1	0/1	Specifies whether taskers should be set to busy upon job dispatch
<i>taskerDisconnResv</i>	timespec	5m00s	[1m00s--1d00h]	Delay in cleaning up the tasker reservation on tasker connection error
<i>taskerheartbeat</i>	integer	60	[1--300]	Default heartbeat for taskers, in seconds. Can be overridden for individual taskers via vovtaskermgr configure or by starting the vovtasker with the -U argument
<i>taskerload.usedb</i>	boolean	0	0/1	If set to 0, use taskerload data from *.SWD/data to generate wave files If set to 1, use taskerload data from time-series database to generate wave files
<i>tasksMaxEffort</i>	integer	20	[5--50]	Percent of time to give to maintenance tasks
<i>thread.preempt.ma</i>	integer	40	[10--200]	Max number of concurrent threads for auxiliary preemption work like vovlmremove

Name	Type	Default	Range	Description
<code>thread.service.en</code>	boolean	1	0/1	Enable safe threads for VovFsGroups
<code>thread.service.en</code>	boolean	1	0/1	Enable safe threads for queries
<code>thread.service.fo</code>	double	0.020000	[0.001000--2.00000]	Reference cost for a fork() used to decide when to fork() for service threads
<code>thread.service.ma</code>	integer	0	[0--50]	Max number of concurrent forked threads for read-only services
<code>thread.service.mi</code>	integer	50000	[0--40000000]	Min Number of nodes in set to activate safe threads
<code>thread.service.mi</code>	integer	500	[0--1000000]	Min Number of resmaps to activate safe threads
<code>thread.service.mi</code>	integer	500	[0--100000]	Min Number of taskers to activate safe threads
<code>timeTolerance</code>	time spec	0	[0,600]	Control the time tolerance of the server. The default value of 0 may be too strict for some networks. You will usually need to have ntp running to synchronize the clocks to use zero, but this is the best practice. Values up to about 3 seconds will not impose

Name	Type	Default	Range	Description
				much performance penalty.
<i>trustHostReported</i>	boolean	0	0/1	Used mostly for demos
<i>trustUserReported</i>	boolean	0	0/1	Check for client user inconsistencies
<i>unsetSkipFlagOnCo</i>	timespec	30s	[1s--24m00s]	Interval in minutes to unset the skip flag on job conflicting with the future reservation
<i>useepoll</i>	boolean	0	0/1	Switch between poll(0, default) or epoll(1) for client servicing (Linux only).
<i>userArtifactClean</i>	timespec	4h00m	[1h00m--1d00h]	Maximum timeout for forcefully cleaning up user-created artifacts that are no longer in use, irrespective of server load
<i>userArtifactClean</i>	timespec	10m00s	[1m00s--1d00h]	Minimum timeout for cleaning up user-created artifacts when the server is under low load or idle
<i>userGroupLowerCas</i>	boolean	0	0/1	Force user group name to be lower case
<i>userLowerCase</i>	boolean	0		If set to 1, force all user names to lowercase. This applies also to ACL and to the

Name	Type	Default	Range	Description
				security modules (vtk_security).
<i>usrtmp</i>	path	/tmp		/tmp, /usr/tmp, /var/tmp, c:/temp
<i>vovIdHigh</i>	integer	999,999,999	[1,000,000 - 2,000,000,000]	When recycling vovId's, this is the largest usable ID. This will be at least 1,000,000 more than vovIdLow.
<i>vovIdLow</i>	integer	5,000	[5,000,10,000,000]	When recycling vovId's, this is the ID that the that generator wraps back to.
<i>vovstring.maxLeng</i>	unsigned	2147483647	[65535--429496729	Max length of a VovString object
<i>vovwxd.arraysizep</i>	boolean	0	0/1	If set, the server will set a property for the final launcher array size when using local resources
<i>vovwxd.fastexit</i>	boolean	1	0/1	Enable/disable vovwxd fastexit
<i>vovwxd.localresou</i>	boolean	0	0/1	If set, the user facing queue will manage local resourcemaps
<i>waitreasons.usedb</i>	boolean	0	0/1	If set to 0, use waitreasons data from *.SWD/data to generate wave files If set to 1, use waitreasons data from time-series database to generate wave files

Name	Type	Default	Range	Description
<code>webport</code>	integer	0	[0--65535]	The web port used by http server (possibly with ssl)
<code>webprovider</code>	string	internal		The facility hosting the web port. Either internal or nginx.
<code>wx.dispwithbucket</code>	boolean	1	0/1	Use Bucket:XXX resource shortcut in Accelerator Plus
<code>wx.setQueueEnv</code>	boolean	0	0/1	Whether to set NC_QUEUE in the job env to Accelerator Plus project.

Troubleshooting the Server

If the server does not start, take the following steps:

1. Check if the server for your project is already running on the same machine.



Note: Do not start a VOV project server more than once.

For example:

```
% vovproject enable project
% vsi
```

2. Check if the server is trying to use a port number that is already used by another vovserver or another application. VOV computes the port number in the range [6200,6455] by hashing the project name. If necessary, select another project name, or change host, or use the variable VOV_PORT_NUMBER to specify an known unused port number. The best place to set this variable is in the `setup.tcl` file for the project.
3. Check if the server is trying to use an inactive port number that cannot be bound. This can happen when an application or the server terminates without closing all its sockets.
4. When a port is not available, the server will exit with a message similar to the following:

```
...more output from vovserver...
vs2 Nov 02 17:34:55 0 3 /home/john/vov
vs2 Nov 02 17:34:55 Adding licadm@venus to notification manager
vs2 Nov 02 17:34:55 Socket address 6437 (net=6437)
```

```
vs52 ERROR Nov 02 17:34:55 Binding TCP socket: retrying 3
vs52 Nov 02 17:34:55 Forcing reuse...
vs52 ERROR Nov 02 17:34:58 Binding TCP socket: retrying 2
vs52 Nov 02 17:34:58 Forcing reuse...
vs52 ERROR Nov 02 17:35:01 Binding TCP socket: retrying 1
vs52 Nov 02 17:35:01 Forcing reuse...
vs52 ERROR Nov 02 17:35:04 Binding TCP socket: retrying 0
vs52 Nov 02 17:35:04 Forcing reuse...
vs52 ERROR Nov 02 17:35:04
PROBLEM: The TCP/IP port with address 6437 is already being used.

POSSIBLE EXPLANATION:
- A VovServer is already running (please check)
- The old server is dead but some of its old clients are still alive (common)
- Another application is using the address (unlikely)

ACTION: Do you want to force the reuse of the address?
```

In this case, do the following:

- a) List all VOV processes that may be running on the server host and that may still be using the port. For example, you can use:

```
% /usr/ucb/ps auxww | grep vov
john 3732 0.2 1.5 2340 1876 pts/13 S 17:36:18 0:00 vovproxy -p
acprose -f - -b
john 3727 0.1 2.2 4816 2752 pts/13 S 17:36:16 0:01 vovsh -t /rtda/
VOV/5.4.7/sun5/tcl/vtcl/vovresourced.tcl -p acprose
...
```

- b) Wait for the process to die on its own, or kill the process with the command `vovkill`. For example:

```
% vovkill pid
```

- c) Restart the server.

If there is trouble connecting to the vovserver, check the canonical name of the server working directory. For information, refer to [Server Working Directory](#).

Change the Project Name, Server Host, or User

Change the Project Name

To change the project name, change the variable `VOV_PROJECT_NAME` in the `setup.tcl` file.

The following example uses a C-shell:

```
% cd `vovserverdir`
% vovproject stop oldname
% mv oldname.swd newname.swd
% vi newname.swd/setup.tcl # Change VOV_PROJECT_NAME
```

```
% vcs newname.swd/setup.tcl.  
% vovproject start
```

Change the Server Host

To change the host of a project, the vovserver must be stopped and the host references fixed in the following files:

- The `setup.tcl` file in the server working directory
- The registry entry (normally located in `$VOVDIR/local/registry/user/`)
- For Accelerator only, the queue-specific configuration file in the `NC_CONFIG_DIR` directory (usually `$VOVDIR/local/vncConfig`)

1. Change to the server working directory.

- a) If the server is still running:

```
% cd `vovserverdir`
```

- b) If the server is not running, find the directory:

```
% vovproject list -l | grep projectName
```

2. If necessary, shut down the server as shown below.

```
% vovproject stop projectName
```

3. In the server working directory, edit the `setup.tcl` file and change the value on the line that sets the environment variable `VOV_HOST_NAME`.

4. Log onto the new host.

5. Restart the project on the new host:

```
% vovproject start projectName
```

6. Clean up the registry. In the directory `$VOVDIR/local/registry/user/`, locate the file `projectName@oldHostName` which can be safely removed. There is another registry entry `projectName@newHostName` which can be kept.

Rehost a Project

To rehost a project, start the project on a different host and pass the `-rehost` option.
For example:

```
% vovproject start -rehost
```

Block a Project from Returning to Shell

The option `-block` is used to prevent the project startup from returning to the shell. This is useful when the project is being started as a batch job in a scheduler environment. The `-block` option can be used when starting or creating a vovproject.

Examples:

```
% vovproject start -block  
% vovproject create -block
```

Change Ownership of VOV Projects

It can be necessary or useful to change the owner of a VOV project. For example, a project that was started under an individual's account, but will be shared and should be owned by a role account instead.

VOV records the initial creator of a project as the owner of the project in the project's regentry file. This enables the `vovproject list` command to show a user the projects that the user has created.

There are several ways change the ownership of a project. Regardless of the method used, it is **essential** to first stop the project's vovserver before making any changes. The vovserver periodically updates the status of a project and re-writes the regentry file; any changes made while the project is running may be overwritten by the vovserver.

The following are two examples for changing project ownership.

Change Ownership, Method 1

1. Run `vovproject stop`, and then run `vovproject destroy`.
This will remove the project's `.swd` and registry entry.
2. Log in as the desired user.
3. Recreate the project's `.swe` and registry entry
This destroys all record of the existence of the project, including the state of the jobs and files in the project's flow, which may be undesirable.

Change Ownership, Method 2

In this example, you'll stop the project, change the OS ownership of its files, and manually change the project metadata recorded by VOV in the regentry file. Follow the steps to do so:

1. Locate the project's configuration (`.swd`) directory with the command `vovproject list -all -l`.
The `.swd` directory is in the directory shown in last column.
2. Change they ownership using OS commands, such as `chown -R` on UNIX or Linux.
3. Edit the `security.tcl` file in the `.swd` to add the new owner as admin.
4. Save the file so that it is owned by the new owner, and is not readable by anyone else.

5. Locate the regentry file in the registry, being careful to get the one for the right VOV version if you are not in a `local` directory that is shared among versions.
6. Open the regentry file and locate the line that looks like:

```
set owner      "previous-owner"
```

7. Change the name between the quotes to the desired login account write and quit.
8. Use `vovproject list -all` to verify it shows the desired owner for the project.
9. Get a shell as the desired owner on the project's vovserver host, and start the project using `vovproject start project-name`
10. Carefully review the result for messages about ownership.

Journals

The vovserver records all events in a journal file that resides in a the subdirectory `journals/` of the server configuration directory.

Each journal has a name in the form `YYYY.MM.DD.jrn`. A new journal is started each day; older journals are compressed automatically, but not removed. For a long-running VOV project, it may be necessary to set up the [vovcontrab](#) or another means to manage the size of the journal's directory.

In the current release, the journals are intended for machine consumption, and are terse and cryptic. These journals are to be used for auditing and troubleshooting.

The journals can be browsed on the Journals page. This page can display only the events that are related to a specific node, or all events. The events are arranged into groups by timeslice.

Log Files

The server generates several daily log and data files, including:

- Server log files in `*.swd/logs`
- Alert log files in `*.swd/logs`
- Journal files in `*.swd/journals`
- Jobs files in `*.swd/data/jobs`
- Resources files in `*.swd/data/resources`
- Taskerload files in `*.swd/data/taskerload`
- Waitreasons files in `*.swd/data/waitreasons`
- Checkouts files in `*.swd/data/checkouts` (this applies only to Monitor)
- Denials files in `*.swd/data/denials` (this applies only to Monitor)

The files are closed after midnight and a new file is opened for the next day with the appropriate name.

It can be desirable to process the log files as they are being created. The log handler is a program that reads the log as it is being created. Typically, the log handler is invoked by substituting two keywords in the command line for the handler:

- @LOGFILE@ is the full path of the log file.
- @EXPIRE@ is the expiration timestamp for the log file, typically midnight of the current day.

A useful log handler is included in VOV, called `vovloghandler`. The usage of the log handler:

```
vovloghandler: This script is an example of a log handler to be used with dailylogs
vovloghandler: It is used to start a process to monitor the logs
vovloghandler: and to process them as they get generated.
vovloghandler: In this example, we use a simple 'tail -f ...'
vovloghandler:
vovloghandler: Usage: vovloghandler @FILENAME@ @EXPIRE@
vovloghandler:
```

The log handler is executed in the background and it must be able to exit by itself. Examples of conditions to terminate a log handler:

- The expiration time has been reached.
- A file called @FILENAME@.closed; @FILENAME@ is substituted with the full path to the log file.

```
#
# This is a fragment of policy.tcl
#
# Example of flags to control which handlers are called for the log files:
#
set config(logs,server,handler)      ""
set config(logs,journals,handler)    ""
set config(logs,jobs,handler)        "vovloghandler @FILENAME@ @EXPIRE@"
set config(logs,resources,handler)   ""
set config(logs,waitreasons,handler) ""
set config(logs,checkouts,handler)   "vovloghandler @FILENAME@ @EXPIRE@"
```

Old logs can be cleaned up using `vovcleanup`.

Compress Log Files

After closing a daily log file, the server can automatically compress the file to save on disk space. The utility to compress the file is `vovcompresslog`, which calls `gzip` or any other utility to compress the files.

```
vovcompresslog: Usage: vovcompresslog <LOGNAME>
```

To control which log files are compressed, edit the `policy.tcl` file and set the `config(log,XXXX,compress)` flags. Example:

```
#
# This is a fragment of policy.tcl
#
# Example of flags to control which logs get compressed.
#
set config(logs,server,compress)      0
set config(logs,journals,compress)    1
set config(logs,jobs,compress)        0
set config(logs,resources,compress)   1
set config(logs,waitreasons,compress) 0
```

```
set config(logs,checkouts,compress) 1
```

Reopen the Logs After an Error

If the logging encounters an error, such as a full disk, logging will be suspended and an alert will be issued. After clearing the full disk condition, restart the logs with the following command:


```
% vovsh -x 'vtk_server_config reopenlogs 1'
```

Start and Configure the Server

Although the Monitor daemons are started automatically upon startup, there may be times where a daemon should be restarted. This section describes how to control the daemons from the command line (CLI).

CLI-based Operation

To control daemons in the CLI, a utility is provided called `vovdaemonmgr`

 **Note:** Daily log files generated by system daemons are now automatically compressed (non-Windows only).

vovdaemonmgr: Usage Message

This is the command to show the status of daemons. You can also use this command to start/stop the daemons.

NOTE: This command can only be used by the owner of the vovserver and on the machine where the vovserver runs.

USAGE:

vovdaemonmgr <SUBCOMMAND> [OPTIONS] [daemonsList]

SUBCOMMAND is one of:

```
list      -- List the configured daemons. (list -all for all daemons)
restart   -- Restart the specified daemons.
show      -- Show the status of the specified daemons.
status    -- Same as 'show'.
start     -- Start the specified daemons.
stop      -- Stop the specified daemons.
```

[daemonsList] is optional. When omitted, act on all daemons.

OPTIONS:

```
-f          -- Force flag (for start only). If the start subcommand
             has an explicit daemonsList, the specified daemons
             will be started even if not configured.
-h          -- Help usage message.
-v          -- Increase verbosity.
-retry <N>  -- For NIS, control how many retries to attempt, default 0
-wait <N>   -- For NIS, control how long to wait between retries, default 0s
```

EXAMPLES:

```
% vovdaemonmgr list
% vovdaemonmgr list -all
% vovdaemonmgr status
% vovdaemonmgr status vovpreemptd
% vovdaemonmgr start vovnotifyd
% vovdaemonmgr start -force vovresourced
% vovdaemonmgr start -force -v -v vovresourced
% vovdaemonmgr stop vovlad
```

Autostart Directory

With the command `vovautostart`, on `vovserver` startup, scripts can be specified to execute automatically.

In UNIX, the scripts can be written in either C-shell or Tcl syntax.

 **Note:** For the script to work in Windows, Tcl syntax must be used. Guidelines follow:

- Create a directory named `autostart` in the server working directory.
- For both UNIX and Windows:
 - Create a script with the suffix `.tcl` in the `autostart` directory.
- For UNIX only, CSH scripts are also supported:
 - Create a script with the suffix `.csh` in the `autostart` directory.
 - Ensure the script has the appropriate executable permissions.

Each script in the `autostart` directory is called with one argument, which is the word `start`. This argument is usually ignored in OEM scripts, but can be used to in custom scripts to enforce different behaviors between a manual call on the CLI versus an automated call by the `vovserver`.

Examples are available in the directory `$VOVDIR/etc/autostart`.

vovautostart

The scripts are launched by the utility `vovautostart`. To repeat the execution of the `autostart` scripts, `vovautostart` can be executed from the command line.

vovautostart: Usage Message

DESCRIPTION:

Execute the scripts in the `*.swd/autostart` directory.

There are three types of scripts that get executed:

1. Scripts that match `*.sh` are executed directly (Unix only)
2. Scripts that match `*.csh` are executed directly (Unix only)
2. Scripts that match `*.tcl` are executed by `vovsh`.

The scripts are executed in alphabetical order in the background, with a 5s a 5s delay between successive scripts.

This utility is normally invoked by `vovserver` upon launching.

USAGE:

```
% vovautostart [optional directory spec]

EXAMPLES:
% vovautostart
```

Run Periodic Tasks with vovliveness

If the directory "tasks" exist in the Server Working Directory, the server calls the `vovliveness` script once per minute.

The script executes all the tasks contained in the "tasks" directory.

```
vovliveness: Usage Message

DESCRIPTION:
  This script is called by vovserver about once a minute.
  It can be used to perform maintenance tasks.

USAGE:
  % vovliveness [OPTIONS] <taskdirectory> <timestamp>

WHERE:
  task_directory      -- is the directory with the tasks
                       to be executed. The tasks are those
                       that match the expression "live_*.tcl".
  timestamp           -- Currently ignored.

OPTIONS:
  -v                  -- Increase verbosity.
```

There are many uses for `vovliveness`. Examples are available in the directory `$VOVDIR/etc/liveness`.

To activate this functionality, create the directory `tasks` and add some tasks files with a name matching the expression `live_*.tcl`. The Tcl interpreter has access to all `vtk_*` procedures. Example:

```
% cd `vovserverdir -p .`
% mkdir tasks
% cd tasks
% cp $VOVDIR/etc/liveness/live_start_taskers.tcl .
```

Following an example of the script `live_start_taskers.tcl` to restart any down taskers, once per hour:

```
#
# Copyright © 2007-2025, Altair Engineering
#
# All Rights Reserved.
#
# Directory : src/scripts/liveness
# File      : live_start_taskers.tcl
# Content   : Start down taskers once an hour.
# Note      :
#
# $Id: //vov/branches/2019.01/src/scripts/liveness/live_start_taskers.tcl#3 $
```

```
#

set now [clock seconds]

# Get or initialize period
if { [catch {set period [vtk_prop_get 1 LIVE_START_TASKERS_PERIOD]}} ] {
    set period 3600
    catch {vtk_prop_set 1 LIVE_START_TASKERS_PERIOD $period}
}

# Get age
if { [catch {set lastRun [vtk_prop_get 1 LIVE_START_TASKERS_LAST]}} ] {
    set lastRun 0
}
set age [expr {$now - $lastRun}]

if { $age >= $period } {

    # Start down taskers
    if { [catch {exec vovtaskermgr start >&@ stdout} errmsg] } {
        VovError "Failed to start taskers: $errmsg"
    }

    # Reset the last run TS
    catch {vtk_prop_set 1 LIVE_START_TASKERS_LAST $now}

}
```

Alerts from Liveness Tasks

Alerts may occur that are related to liveness tasks such as "The previous liveness script is still connected", especially in Monitor.



Note: In previous releases, there is no control these occurrences; such occurrences cause no harm.

The liveness tasks system is designed to support short jobs that are triggered frequently (about once per minute) by the vovserver so long as it is running. It was also used for the database loading task for Monitor checkouts and Accelerator jobs; sometimes these jobs run significantly longer.

In later releases the debuglog parsing, batch reports and other maintenance items are converted to periodic jobs that run on a dedicated vovtasker named 'maintainer', so these alerts should no longer appear.

Run Periodic Tasks with vovcrontab

The UNIX utility `crontab` is used to perform regularly scheduled tasks such as retracing an entire project each night or storing a back-up of the trace every Saturday. `vovcrontab` is a VOV utility that simplifies the creation of cron rules for a project. Directions are provided in this section.

Usage: voncontrab

```
vovcrontab: DESCRIPTION:
vovcrontab:      Interface to the UNIX utility crontab.
vovcrontab:
```

```
vovcrontab: USAGE:
vovcrontab:      % vovcrontab [option]
vovcrontab:
vovcrontab: OPTIONS:
vovcrontab:  -help           -- Get this message
vovcrontab:  -new           -- Install the crontab for this project
vovcrontab:                  Also used to update the scripts/vovdir.csh script.
vovcrontab:  -noautostart  -- Do not install autostart script to update
vovcrontab:                  vovdir.csh; the default is to install it.
vovcrontab:  -reinstall   -- Reinstall current crontab for this project
vovcrontab:  -clear       -- Clear the current crontab
vovcrontab:  -show        -- Show the crontab
vovcrontab:  -type <type> -- Specify project type
vovcrontab:
vovcrontab: NOTE:
vovcrontab:   Please remember to copy
vovcrontab:   $(VOVDIR)/etc/autostart/update_crontab_vovdir.csh
vovcrontab:   into your autostart directory if needed.
vovcrontab:   It is installed by default.
```

Enable a Project

Enable a project in a shell via:

```
'vovproject enable <PROJECT>'
```

Create crontabs

Execute 'vovcrontab -new' to create crontabs.

```
% vovcrontab -new
vovcrontab: Creating vnc.swd/scripts/cron.csh
vovcrontab: Creating cron table vnc.swd/crontab.lion
no crontab for john
vovcrontab: Installing new crontab.
vovcrontab: Installing updated crontab
```

This program prepares the scripts \$SWD/scripts/cron.csh and \$SWD/crontab.hostname.

Display Current crontab Definition

Running vovcrontab -show shows the current crontab definition.

```
% vovcrontab -show
#### (vovcrontab) START PROJECT vnc ####
#
# ... some lines omitted...
#
# Every hour at 5 minutes before the hour.
55 * * * * /home/john/vov/vnc.swd/scripts/cron.csh hourly
#
# Every day: at 23:15
15 23 * * * /home/john/vov/vnc.swd/scripts/cron.csh daily
#
# Every week: on Saturday at 7:00am
0 7 * * 6 /home/john/vov/vnc.swd/scripts/cron.csh weekly
#
# Every month: on the 1st at 3:00am
0 3 1 * * /home/john/vov/vnc.swd/scripts/cron.csh monthly
```

```
#### (vovcrontab) END PROJECT vnc ####
```

Customize the crontab

The crontab can be customized by editing either `$SWD/crontab.hostname` or `$SWD/scripts/cron.csh`. Afterwards, `vovcrontab -reinstall` will need to be run to take the modifications into consideration.

```
% vovcrontab -reinstall
vovcrontab: vnc.swd/scripts/cron.csh exists already.
vovcrontab: vnc.swd/crontab.lion exists already.
vovcrontab: Installing new crontab.
vovcrontab: Installing updated crontab
```

Delete the Current crontab Definition

To delete current crontab definitions, use:

```
'vovcrontab -clear'
vovcrontab: vnc.swd/scripts/cron.csh exists already.
vovcrontab: vnc.swd/crontab.lion exists already.
vovcrontab: Removing the crontab
```

Complete the Cleanup

To complete the cleanup, remove the `crontab.HOSTNAME` file in the SWD directory of the project.

```
% rm crontab.lion
```

Job Status Triggers

The daemon `vovtriggerd` taps the event stream and executes commands that are based on selected events.

A typical application is updating an external SQL database when a job is completed.

Triggers are different from post-commands. Triggers are executed by `vovtriggerd`, which is normally run by the user who owns `vovserver`. The owner of the account that was used to start `vovserver` is the *owner* of `vovserver`. Post-commands are executed by the user who owns each job.

The following table summarizes the information about `vovtriggerd`:

Config file	<code>vnc.swd/vovtriggerd/config.tcl</code>
Sample config file	<code>\$VOVDIR/etc/config/vovtriggerd/config.tcl</code>
Info file	<code>vnc.swd/vovtriggerd/info.tcl</code>

Set Up vovtriggerd

`vovtriggerd` is a daemon that is configured as follows:

- Create a subdirectory called `vovtriggerd` in the server configuration directory

- Create a configuration file called `config.tcl` with the main purpose of overriding the procedure called `triggerCallBack`
- Start the daemon:

```
% mkdir `vovserverdir -p vovtriggerd`  
% cd `vovserverdir -p vovtriggerd`  
% mkdir autostart  
% cp $VOVDIR/etc/config/vovtriggerd/config.tcl .  
% vovdaemonmgr start vovtriggerd
```

The TRIGGER Property

The default trigger handler looks for the property `TRIGGER` attached to the object mentioned in the event. If the property exists, it is assumed to be the name of a trigger procedure to be called. There are three arguments for the trigger procedure: `id`, `subject`, `verb`.

The trigger procedures are defined in the `config.tcl` file. Following are the guidelines for implementing a trigger:

- The trigger is stateless.
- The trigger is fast; it should complete within a few seconds.

Following is an example of using the `TRIGGER` property:

- Create a trigger call-back in the `config.tcl` file. In the following example, it is named `trigShowJobEventCB`.

```
#  
# This goes in PROJ.swd/vovtriggerd/config.tcl  
#  
proc trigShowJobEventCB { id subject verb } {  
    puts "TrigShowJobEventCB: Just got the event $id $subject $verb"  
}
```

- Attach the property `TRIGGER` to jobs in the flow. Example:

```
% vovprop set -text 000123456 TRIGGER trigShowJobEventCB  
% vovprop set -text 000234567 TRIGGER trigShowJobEventCB
```

Trigger Events

Following are the events that are processed by `vovtriggerd`:

- JOBID "JOB" "DISPATCH", when the job is dispatched to a tasker.
- JOBID "JOB" "ERROR", if the job fails.
- JOBID "JOB" "STOP", if the job succeeds.

Handling the OVERFLOW Event

If the `vovtriggerd` daemon receives an overflow event (the verb is the string `OVERFLOW`), the procedure `overflowCallBack` is called with no arguments. The overflow event is an indication of a buffer overflow inside the `vovserver`, which is typically caused by `vovtriggerd` being too slow in processing the events. Depending on the situation, it may be useful to reinitialize the trigger callbacks.

Example of Submission of Jobs with Triggers

The following example applies to Accelerator:

A trigger can be submitted by setting the `TRIGGER` property. Knowing the name of the trigger callback routine to call is required. In the following example, the name of the trigger callback is `updateDbCallBack`.

```
% nc run -P TRIGGER=updateDbCallBack sleep 10
```

```
# Example of updateDbCallBack
# This procedure is defined in *.swd/vovtriggerd/config.tcl
# Here we update a table called "mytable" based on the
# value of a property called MYPROP.
proc updateDbCallBack { jobid subject verb } {
    switch $verb {
        "STOP" - "ERROR" {
            if [catch {set value [vtk_prop_get $jobid "MYPROP"]} {}] {
                set value -1
            }

            set stmt "INSERT INTO mytable (id,value)"
            append stmt " VALUES ( $jobid, $value)"
            VovSQL::init
            set handle [VovSQL::open]
            VovSQL::query $handle $stmt
            VovSQL::close $handle
        }
    }
}
```

Sanity Check for vovserver

The command `sanity` is used to perform checks on the consistency of the trace and of other internal data structures.

Use sanity check when the server appears confused about the status of the trace.

```
% vovproject sanity
```

Use the `reread` command to re-read the server configuration. The files read are `policy.tcl`, `security.tcl`, `equiv.tcl`, `setup.tcl`, and `exclude.tcl`.

You need not use `reread` after changes to `taskers.tcl`, it is not a vovserver config file. It is used by `vovtaskermgr`.

```
% vovproject reread
```

`sanity` does a wide variety of checks, cleanups, and rebuilds of internal data structures. Check the vovserver log file for messages that include `sanity`. Here are some of the main things that it does:

- Clears all alerts
- Flushes journal and crash recovery files

- Clears IP/Host caches
- Stops and restarts resource daemon (`vovresourced`)
- Checks and cleans internal object attachments
- Verifies all places and jobs have sensible status
- Resets user statistics and average service time
- Checks the contents of system sets like `System:jobs`
- Removes older jobs from recent jobs set
- Makes sure all jobs in the running jobs set are actually running
- Verifies all sets have the correct size
- Clears the barrier-invalid flag on all nodes and recomputes it
- Clears empty retrace sets
- Checks preemption rules
- Checks all tasker machines, marking them sick if they are not responding
- Checks for rebooted tasker machines and terminates jobs attached to them
- Checks filesystems on tasker machines and verifies mount points
- Clears resource list caches from jobs
- Clears and rebuilds job class sets
- Creates limit resources for ones that are missing
- Verifies grabbed resources (non running jobs should not have any)
- Makes sure only running jobs have stolen resources
- Reserves resources for all running jobs
- Create any missing resource maps for groups and priorities
- For each job with I/O, makes sure outputs are newer than inputs
- Makes sure any file with running status has an input job with running status
- Verifies the status of all nodes
- Checks for stuck primary inputs (primary inputs should only be `VALID` or `MISSING`)
- If a file is invalid or missing, but the input job is `VALID`, turn the job `INVALID`
- Finds running jobs without tasker and changes the status to `SLEEPING`
- Makes sure all input files of a `VALID` job are also `VALID`
- Makes sure all output files of a job have the same status as the job
- Recomputes waitreason counts
- Checks job queue buckets
- Verifies link between job queue buckets and resource maps
- Makes sure all queued jobs have job queue buckets
- Checks FairShare groups
- Checks for a license

Connect to a Project

Connect to a Project via the Command Line

To connect to a VOV project, get a list of the projects with `vovproject list`, and then enable the selected project.

Example:

```
% vovproject list
...
...
% vovproject enable mytestmac05 mytest@mac33 DEFAULT documentation/html >
```

After connecting to a VOV project, a different prompt may appear, which indicates for the current project and the current environment.

The project is now set up for interaction. Example:

```
% vsi
% vovconsole &
% vsr -all
```

Alerts

VOV issues an "alert" when an event requires attention. An alert can range from information that does not require action to an urgent fault that requires immediate action.

Depending on the event that occurs, an alert may require attention from a system administrator.

VOV supports four alert levels, which are defined in the table below.


INFO	Information only, no action required.
WARN	Warning, a limit is about to be reached.
ERROR	A fault in one of the subsystems. Example: a syntax error in one of the configuration files.
URGENT	A major fault that compromises the behavior of the system and requires immediate attention. Examples: a license violation or a disk full condition.

Alerts can be viewed on the browser or the command line interface (CLI). In addition, alerts are stored in log files.

For the administrator, VOV permits two actions with respect to an alert:


- Acknowledge the alert.

- Delete the alert from view.

 **Note:** Every alert is stored in a log file; deleting an alert from viewing does not delete the record of the alert in the log file.

Maximum Number of Alerts

The vovserver keeps up to a defined maximum number of alerts in view. The maximum number is defined by the parameter `alerts.max`. The default value is 50. If the number of alerts exceeds the maximum, the oldest alert with the lowest level is deleted from the view.

 **Note:** The record of the alert is not deleted from the log file.

Manage Alerts

For viewing alerts, the level of the most severe alert is visible in the title bar of the browser user interface and in the VOV GUI. The most recent alerts can be viewed from the command line interface with the following commands:

- `vsi` for short format
- `vovshow -alerts` for full format

The following is an example of alerts as shown by `vsi`:

```
Alerts:
URGENT      Imminent license violation: us      3      12d20h      12d20h
WARNING      License is expiring in less th      2613    12d20h      9d13h
ERROR        License is expiring in less th      558     9d13h      8d13h
URGENT      License has expired                    270     8d13h      8d01h
URGENT      License violation                    24      8d13h      8d01h
WARNING      License violation: too many sl      336     8d13h      8d01h
```

The following example shows the same alerts as seen on the browser:

#	Level	Module	Title	Occurrences			Actions
				Count	Earliest	Latest	
1	WARNING	vovresourced	Missing actualTags list for a feature. Feature EDA/al40db++, while updating tags RTDA_RLM1	304282	20d13h	10s	[del] [ack]
2	INFO	vovdaemonlib	Cannot start vovnetappd couldn't execute "vovnetappd": no such file or directory	494	20d13h	15m16s	[del] [ack]
3	INFO	vovdaemonlib	Cannot start vovsplunkd couldn't execute "vovsplunkd": no such file or directory	494	20d13h	15m14s	[del] [ack]
4	WARNING	vovnotifyd	74 configured tasker(s)/agent(s) not running. Use the taskers page to restart taskers that are down	1080	7d15h	8m42s	[del] [ack]
5	WARNING	vovresourced	Failed to get a reply from lm.int.rtda.com:5555 /raw/licdaemons, SSL true	1	21h48m	21h48m	[del] [ack]
6	URGENT	vovresourced	Monitor is unreachable Check that Monitor is running at 5555@lm.int.rtda.com,licmon, see vovresourced log for details	1	21h48m	21h48m	[del] [ack]

Figure 4:

If viewing the documentation from a live vovserver, refer to the Alerts page.

Alerts are also logged in the logs directory in files with names that are formatted as `alert.YYYY.MM.DD.log`. Old alert files are compressed.

Some alerts may not require immediate action. However, it is good practice to acknowledge the alert. The [ack] link on the alerts page can be used to indicate that the alert has been acknowledged. The login name of the person acknowledging the alert will be shown on the Alerts page.

Clear Alerts from View

An alert is automatically cleared from view about one day after the last occurrence that triggered the alert. A selected alert can be removed from view by using the [del] link from the browser UI.

All alerts can also be cleared from view with the following command:

```
% vovforget -alerts
```

Tcl API

There are two Tcl API procedures in `vovsh` that handle alerts:

- `vtk_generic_get alerts A` - Get alert data into array A
- `vtk_alert_add sev title` - Add an alert

To add an alert from the Tcl interface, use the command `vtk_alert_add`.

To get data for all alerts in Tcl, use the command

```
vtk_generic_get alerts array
```

The following code example shows how `vsi` formats the alerts:

```
# This is how the vsi cmd formats alerts
vtk_generic_get alerts alerts
if { $alerts(count) > 0 } {
    append output "\nAlerts:\n"
    for { set i 0 } { $i < $alerts(count) } { incr i } {
        append output [format " %-10s %-10s %-30s" $alerts($i,level)
                             [string range $alerts($i,module) 0 9]
                             [string range $alerts($i,title) 0 29] ]
        if { $alerts($i,count) > 1 } {
            append output [format "%7d %8s %8s"
                                $alerts($i,count)
                                [vtk_time_pp [expr $now - $alerts($i,first)]]
                                [vtk_time_pp [expr $now - $alerts($i,last) ]]]
        }
        append output "\n"
    }
    append output "\n"
}
```

Thin Clients

A *thin client* (VOV protocol) runs on a machine that does not have access to the SWD directory of the vovserver.

Connect a Thin Client

Such a client cannot read the configuration files for the project, such as the `equiv.tcl` file that defines logical filesystem names, or the `setup.tcl` file that defines necessary server environment variables.

An example of a thin client is the `vovinfo` binary, a stripped-down version of `vovsh` that is used for host monitoring in Monitor.

Environment Variables

To use a thin client, it is recommend to set variables as shown below:

```
setenv VOVEQUIV_CACHE_FILE vovcache
setenv VOV_SWD_KEY none
```

This setting of `VOVEQUIV_CACHE_FILE` disables checking the `equiv.tcl` file. The client gets equivalences from the server cache, if available, via the VOV protocol RPC. Setting `VOV_SWD_KEY` to `none` causes the client to avoid accessing the server working directory.

Troubleshooting: Cannot Enable Project

Problem: you run the following command:

```
% vovproject enable MYPROJECT
```

and you receive an error similar to this:

```
vovproject 06/16/09 10:53:47: FATAL ERROR: Cannot find project 'MYPROJECT for user 'john'
```

There may be other projects that are in conflict with the current project, perhaps running on the same machine and have the same project name, but a different port number.

Troubleshooting tips:

```
% vovproject list -a
```

- Is your project listed? Is it running?
- Are there multiple projects with the same name?

Perhaps the VOV registry is corrupted for this project - check the `$VOVDIR/local/registry`.

- Is there a file similar to `MYPROJECT@SOMEHOST` ?
- Are there more files like that?

A safe mode to enable the project is with the `ves` command:

```
% source $VOVDIR/etc/vovrc.csh      ## To make sure the alias is loaded.
% ves path/to/mytestproject.swd/setup.tcl
```

Security

The VOV security model consists of assigning a *Security Code* to each client or client group (VovUser Group) and granting the permission to execute critical tasks only to clients with the appropriate security level.

Refer to [VovUserGroups](#) for more information about how VovUser Groups are created and maintained.

Security Levels

There are five different security levels.

The security levels are:

Security Name	Numeric Level	Description
ANYBODY	1	No privilege to run any CLI command.
READONLY	2	Minimum privileges; a user can only browse the information but cannot change anything
USER	3	<p>A user can only execute established flows and view non critical information.</p> <p>In particular, a USER:</p> <ul style="list-style-type: none"> • May create, modify or forget his own jobs • May create, modify or forget his own files • May create, modify or forget his own set • May modify or forget his dependencies • Create/modify/start/stop/forget own tasker • May not forget jobs owned by other users • May not modify jobs owned by other users • May not start and stop taskers • May not stop the server
LEADER	4	<p>Intermediate privileges; a user can create and execute arbitrary flows and view all non-security related information.</p> <p>A LEADER:</p>

Security Name	Numeric Level	Description
		<ul style="list-style-type: none"> • May start or stop his own taskers • May forget all jobs, including jobs owned by another user • May save trace DB to disk <p>This level is rarely used.</p>
ADMIN	5	<p>Administrator privileges; a user has access to most security information</p> <p>An ADMIN:</p> <ul style="list-style-type: none"> • May forget jobs owned by other users • May stop jobs owned by other users • May not modify jobs owned by another user (no user can modify another user's jobs) • May stop the server • May stop/modify/forget the taskers • May refresh tasker cache • May destroy user • May destroy host • May create or destroy alerts • May create, modify or destroy resource map • May reserve resource • May create, modify or destroy preemption rule • May create, modify or destroy multi-queue objects (Monitor sites, Accelerator queues, resources) • May create, modify or destroy Monitor objects (licdaemons, features) • May not modify jobs owned by another user (because no user can modify another user's jobs)

The Security File

The file `security.tcl` in the server working directory specifies the security policies. This file must be owned by the project owner and must have read/write access only for the owner.

To change the security file of an active server, use the `vovproject reread` command to make the changes effective.

The security policies are defined by the Tcl procedure `vtk_security`. A summary of the Tcl procedure follows:

```
vtk_security username|-group vovusergroupsecurityLevelhostNameOrIpRange ...
```

where:

username

The login name of a user or + to mean "anybody" or - to mean "nobody".

vovusergroup

The name of a [VovUserGroups](#).

securityLevel

USER, LEADER or ADMIN (case insensitive)

hostListOrIpRange

For an IP range, it must have the form "x.x.x.x-y.y.y.y" (example 192.168.10.220-192.168.10.240). If it is not an IP range, it is either the name of a host or + to indicate "any host" or - to mean "no host".

Placing the order of the rules in this file is not important. The rules are automatically reordered from the most user specific to the least user specific and from the most liberal to the most restrictive with respect to the security level.

Example: least restrictive security

The least restrictive security grants every user full access from any host.

```
# All users (+) are administrators from all hosts (+).  
vtk_security + ADMIN +
```

Example: most restrictive security

```
# No rule defined gives only the owner of the project ADMIN privileges  
# on the server host.
```

Example: typical case

The following example shows a typical security file, in which different privileges are granted to different users. Note the use of variables in the following example:

```
set servers      { reno milano }  
set allhosts     { reno milano elko tahoe }  
  
vtk_security mary      ADMIN  +  
vtk_security john      ADMIN  tahoe  
vtk_security dan        ADMIN  $servers  
vtk_security pat        LEADER elko  
vtk_security fred       USER   $allhosts  
vtk_security bob        ADMIN  192.168.0.30-192.168.0.100  
vtk_security -group mygroup USER  $allHosts
```

In the example, `mary` is an administrator from any host, and `dan` is an administrator only from `reno` and `milano`. The user `pat` is a LEADER from her machine `elko`, and `fred` has USER privileges with the

four machines that are defined in the variable `$allhosts`. Additionally, members of the `VovUserGroup "mygroup"` have USER privileges from `$allHosts`.

Find the Security Level

To find the security level, use `vtk_user_security` from a Tcl script.

In a CGI environment, the variable `SECURITY_LEVEL` contains the numeric level of security. Example:

```
% vovsh -x "puts [vtk_user_security]"
5 ADMIN
% env SECURITY_LEVEL=3 vovsh -x "puts [vtk_user_security]"
3 USER
```

Operations by Security Level

The VOV security model consists in assigning a *Security Code* to each client (browser, GUI, CLI) and to grant the permission to execute tasks only to clients with appropriate security level.

VOV defines four ordered privilege levels, shown from the least privileged to the most privileged:

- READONLY
- USER
- LEADER
- ADMIN

VOV security is enforced by the server process. Each time that a client requests a transaction, the security level of that client's owner is compared to the definitions in the project's `security.tcl` file, and permission is granted or denied accordingly.

The following table shows the operations permitted to clients according to their privilege level. 'Y' indicates the operation is permitted, otherwise it is not.

VOV Operations allowed by Privilege Level						
Object	Operation	Description	Privilege Level			
			READONLY	USER	LEADER	ADMIN
Trace						
	ViewStatus	View status information about jobs and files	Y	Y	Y	Y
	CreateJob	Add job to flow you own		Y	Y	Y

VOV Operations allowed by Privilege Level						
Object	Operation	Description	Privilege Level			
			READONLY	USER	LEADER	ADMIN
Job						
	RunJobSelf	Run a job you own		Y	Y	Y
	RunJobOther	Run a job owned by another		Y	Y	Y
	StopJobSelf	Stop a job you own		Y	Y	Y
	StopJobOther	Stop job owned by another			Y	Y
	ModifyJobSelf	Modify a job you own		Y	Y	Y
	ModifyJobOther	Modify anything other than legal exit status or resources of job owned by another				
	ModifyJobExit	Modify legal exit status for job owned by another				Y
	ModifyJobResource	Modify resource requested by job owned by another				Y
	ForgetJobSelf	Forget a job you own		Y	Y	Y

VOV Operations allowed by Privilege Level						
Object	Operation	Description	Privilege Level			
			READONLY	USER	LEADER	ADMIN
Project	ForgetJobOther	Forget a job owned by another			Y	Y
	StartTasker	Start a vovtasker of this project			Y	Y
	StopTasker	Stop a vovtasker of this project		Y	Y	Y
	StartServer	Start this project's server				Y
	StopServer	Stop this project's server				Y
	ViewSecurity	See project security info				Y
	ModSecurity	Change project security info				Y

Protection from DoS Attacks

The vovserver implements several protections against denial-of-service (DoS) attacks. One of these is the detection of clients that use a relatively large share of the server time. These clients are considered *service hogs*, and their subsequent service requests can be ignored for up to 3 seconds.

The statistics for server time usage are reset every 10 minutes.

The only evidence of this protection taking place is information in the server log file. The information would look similar to the following:

Service hog: user john 3.2 > 3.0 longest=0.9 penalty=2

The penalty is the number of seconds that the next service will be delayed.

VovUserGroups

VovUserGroups are an access control mechanism. They are internally-defined groups of users that are utilized to make the job of allocating access easier within Altair Accelerator tools.

VovUserGroups by themselves do not grant or restrict access. Once the group names are defined, however, they can be assigned security roles and thus make the job of administering security and access less time consuming.

The `vovusergroup` utility is used to manage *VovUserGroups* from the command line. This includes creating and deleting groups, appending them, and displaying group members.

VovUserGroups Utilization

VovUserGroups can be effectively utilized in a number of areas throughout the Altair Accelerator product suite. In all Altair Accelerator products, they can be used to easily set up access by mapping a *VovUserGroup* to a defined security role in the `security.tcl` file that resides in the Server Working Directory (SWD). *VovUserGroups* can also be used to assign access or restrictions to various components of the web interface.

Usage of `vovusergroup` utility:

```
vovusergroup: Usage Message
```

```
Utility to show, create and maintain "vovusergroup" structures based on  
user lists, Unix groups and LDAP groups.
```

```
USAGE:
```

```
    vovusergroup <ACTION> <GROUP_NAME> <OPTIONS>
```

```
ACTIONS:
```

populate	Creates a new vovusergroup and populates it from a provided userList, an existing unix group or LDAP group. Any pre-existing vovusergroup with the same name is lost.
addusers	Creates a new vovusergroup (if necessary) and appends the provided userList.
removeusers	Removes the specified "userList" list of users from an existing vovusergroup.
delete	Removes a vovusergroup definition entirely, along with its list of users.
show	Displays the list of users in an existing vovusergroup.
list	Displays the defined vovusergroup names

```
SYNTAX:
```

```
vovusergroup populate    <groupName> <userList>  
vovusergroup populate    <groupName> -unix <unixGroup>  
vovusergroup populate    <groupName> -ldap <ldapGroup>
```

```
vovusergroup addusers    <groupName> <userList>
vovusergroup removeusers <groupName> <userList>

vovusergroup delete      <groupName>
vovusergroup show        <groupName>
```

ABOUT USER LISTS:

A "userList" above consists of usernames separated by spaces.
To generate a vovusergroup from LDAP, you must first have LDAP
connectivity enabled (see Altair Monitor Administration Guide)

EXAMPLES:

```
vovusergroup populate chipA -ldap chipA_govusers
vovusergroup addusers myBlk jimc terryw suep ronaldb
vovusergroup show myBlk
```

VovUserGroups Support in Altair Accelerator Products

Currently, support for VovUserGroups is provided in the `security.tcl` file (in all products), and the `web.cfg` file (in Monitor only).

Stop the Server

Shut Down a Project

Any user with ADMIN privileges can shut down a project. However, only the project owner can restart it.

To shut down a project's vovserver, use one of the following methods. If some project data are missing (registry entry, `setup.tcl`, etc.) see 'Last-ditch shutdown' below.

- From the GUI, select **Project > Close console and stop project**
- From the CLI, use either `% vovproject stop project` or

```
% vovproject stop          ; ##  
% vovstop -project         ; ## New as of 8.2.1  
% vovleader -K server      ; ## (last-resort below)
```

After confirmation, the server executes the shutdown procedure and exits. The shutdown procedure may take a while, depending on the size of the trace and the speed of storage, while the vovserver is saving its data.

- From the browser interface, visit the `/admin` page and select the **Shut Down** link, confirming when prompted.

It is important to use one of these procedures to shut down vovserver cleanly. This releases the license resources and saves the flow information into the database so it will be up-to-date when restarting the project.

All of the data associated with the project is preserved, which enables restarting the project with `vovproject start`

To eliminate the VOV data associated with the project, refer to [Destroy a Project](#). Destroying a project removes its registry entry and `<project>.swd` directory.

Last-resort vovserver Shutdown Method

If some of the project setup data has been lost or removed, standard methods will not work to shut down vovserver cleanly. In these cases, enter the project environment, and use the `vovleader -K server` command to shut down vovserver.

- Registry entry absent (`$VOVDIR/local/registry/project@host`)

If the project's registry entry is missing, you can not use regular `vovproject enable` to enter the project context.

If the `<project>.swd/setup.tcl` file is present, you can enable the project this way:

```
% cd /path-to-dir/project.swd  
% ves ./setup.tcl
```

- No `setup.tcl` file (`.../path-to/<project>.swd/setup.tcl`)

If the project's `setup.tcl` file is absent, you can shut down vovserver cleanly by setting the environment variables manually.

```
+ use 'ps -aef' command to determine vovserver owner if needed
+ use 'netstat' command or /proc on Linux to determine vovserver port
+ get shell on vovserver host as project owner

% setenv VOV_HOST_NAME localhost
% setenv VOV_PROJECT_NAME project-name
% setenv VOV_PORT_NUMBER NNNN
  (the TCP/IP port on which vovserver listens)

% vovleader -K server
  (answer 1 to stop vovserver, 2 to cancel)
```


Automated Script Calls for vovserver Shutdown

Using the autostop facility, scripts can be automatically called upon vovserver shutdown. To enable this feature:

- In the server working directory, create a directory named `autostop`.
- Create a Tcl script within that directory.

Additional information:

- csh scripts are supported in UNIX.
- This feature can be called in CLI with the `vovautostop` utility.

 **Note:** This feature is similar to the `autostart` facility that is called upon vovserver startup.

Destroy a Project

A project must be stopped before it can be destroyed.

When a project is destroyed:

- The registry entry for the project is deleted.
- The server configuration directory is deleted.

To destroy all of the data associated with a particular project, use the following interactive command:

```
% vovproject destroy project
You are about to destroy all files
for project 'project'
running in '/home/john/vovadmin'
Do you really want to do it? (yes/no) yes
vovproject: message: Deleting /home/john/vovadmin/project.swd/
vovproject: message: Deleting registry entry project@host
```

If you are absolutely sure of what you are doing, you can use the option `-force` as in:


```
% vovproject destroy -force project
```

Recovery and Redundancy

Failover Server Candidates

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

The configuration instructions follow.

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

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

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

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

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

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

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

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

```
% vovautostart
```

For example:

```
% vovtaskermgr show -taskergroups
ID          taskername  hostname  taskergroup
000404374   localhost-2  titanus   g1
000404375   localhost-1  titanus   g1
000404376   localhost-5  titanus   g1
000404377   localhost-3  titanus   g1
000404378   localhost-4  titanus   g1
000404391   failover     titanus   failover
```



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

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

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

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

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

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

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

How vovserver Failover Works

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

Tips for Configuring Failover

Following are tips for failover configuration:

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

Migrating vovserver to a New Host

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

```
ncmgr rehost -host NEWHOST
```

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

Crash Recovery Mode

Crash Recovery Mode is activated the next time the server is restarted, if the server was not shut down cleanly. Crash Recovery Mode is part of the Failover Server capability of VOV, which is mainly used in Accelerator. This capability allows VOV to start a new server to manage the queue of a server which has crashed unexpectedly.

When you shut down an VOV Subsystem instance cleanly using the `ncmgr stop` command, the server will save its database to disk just before exiting. When the server is restarted, it will read the state of the trace from disk, and immediately be ready for new work.

Sometimes the vovserver will be stopped unexpectedly, such as due to a hardware problem like a machine crash or memory exhaustion. In such cases, the server will not have a chance to save the project database before terminating.

- In VOV, the main concern is usually the state of the trace, which stores the status all the jobs in your project.
- In Accelerator, there is no trace, and the important thing to preserve is the state of the queue, so jobs do not lose their position and need to be re-queued in the case of a server crash.

Journal Files

The vovserver keeps crash recovery journal files of the events that affect the state of the server. These 'CR' files are flushed whenever the trace data are saved to disk. During crash recovery, the vovserver first reads the last saved state of the trace from the disk data, then applies the events from the CR files.

Crash Recovery Restart

When the server is next restarted after such a crash, the server enters what is called Crash Recovery Mode, which usually lasts about two minutes, but may take longer if the CR files are very large. During this period:

- The server waits for vovtaskers with running jobs to reconnect.
- No jobs are dispatched.
- The server does not accept VOV or HTML TCP connections from `vovsh` or browser clients.
- At the end, the server performs a global sanity check.
- A `crash_recovery_report <timestamp> logfile` is written. It logs any jobs lost by the crash recovery sequence.

If the server is properly shut down, the next time it restarts, it will not enter Crash Recovery Mode, and will be immediately functional.



Note: If the sanity check command cannot connect, the server is still recovering its state from the DB and CR files. Check the size of the `vnc.swd/trace.db/CR*` files.

Example commands:

```
% vovproject enable <project-name>  
% vovproject sanity
```

Files, Equivalences, Exclusions

The files used in a project reside in one or more directories spread all around your filesystems. Each logical name defined in the `equiv.tcl` file implicitly declares the root of a workspace. You can browse the workspaces using the browser interface.

Canonical and Logical File Names

VOV clients and server exchange dependency information by using file names; each file needs a single name that is valid on both the client and the server.

It may seem that each file, could use its full path as its unique name. However, a file may and will have more than one name for the following reasons:

- Links, both hard and symbolic, allows multiple full paths for the same file.
- For any file, it is possible to generate an infinite number of full paths by using the "dot" and "dot-dot" notation (for example, `/usr/bin/ls` can also be written as `/usr/../../usr/bin/./ls`).
- The same file may have different full paths on different hosts due to how the file systems are network mounted.

Canonical Names

VOV defines the *canonical name* of a file to be the full path obtained by removing all symbolic links and all "dots".

In this example, a file system contains the following link:

```
/users/john/projects --> /sandbox/projects
```

With the relative path `~/projects/vhdl/vtech/../../syn/vtech.v`, with respect to the user `john`, the following transformations would apply:

The non-canonical path	<code>~/projects/vhdl/vtech/../../syn/vtech.v</code>
after tilde expansion becomes	<code>/users/john/projects/vhdl/vtech/../../syn/vtech.v</code>
after removing the symbolic link becomes	<code>/sandbox/projects/vhdl/vtech/../../syn/vtech.v</code>
by removing the double dot becomes canonical	<code>/sandbox/projects/vhdl/syn/vtech.v</code>

Logical Names

A canonical name is then turned into a *logical name*. A logical name is one in which the file name begins with the value of a variable.

For example, the name `${HOME}/foo.c` is logical, while `/users/home/john/foo.c` is not.

The use of logical names is critical because the value of the variable used in the name is allowed to be different on different hosts. This is to account for the different ways the file systems are mounted across the network.

For example, the variable `${HOME}` may point to `/users/home/john` on a UNIX machine and to `h:/john` on a Windows NT machine.

All filenames in VOV are logical and canonical names. The logical names are formed according to the rules defined in the [equiv.tcl](#) file.

There are two further advantages in using logical canonical names:

- The average length of names is reduced, which reduces the storage requirements for the trace.
- The trace can be easily moved from one file system to another.

Databases

Design files may reside in different databases. The most common database is the filesystem. VOV offers a generic database interface in the sense that it traces dependencies between named entities residing in persistent databases.

VOV requires a method to obtain the timestamp of each named entity in the database. For example, the most common database is called "FILE", its entities are files, and the method to obtain the timestamp of a file is the OS call `stat()`.

VOV supports multiple databases. Some are supported internally. Others must be supported externally by special clients called database proxies, which provide the server with the needed timestamp for the entities in a database.

Check the Databases page to see the list of databases supported by the current project.

Internally Supported Databases

The following databases are currently supported internally:

FILE

The entities are files. The names are logical canonical path names. The timestamp is the modification time of the file. If a file does not exist, the timestamp is 0.

FILEX

The entities are files, but in this case only the existence of the file, is important; the timestamp is not important. The names are logical canonical path names. If the file exists, the timestamp is fixed in the distant past. If a file does not exist, the timestamp is 0.

To specify a file where you just care about the existence of the file, regardless of its timestamp, the syntax is different whether the file is an input or an output:

	FDL	Instrumentation
Input	I -db FILEX myfile D FILE	VovInput -db FILEX myfile

	FDL	Instrumentation
Output	O -db FILEX - ignore_timestamp myfile D FILE	VovOutput -db FILEX - ignore_timestamp myfile

The change of database is persistent. Next Inputs & Outputs will use the FILEX database, unless you add a -db FILEX or a D FILE line in your flow.

GLOB

The entities in this database are "glob expressions." The syntax is the one used by the glob procedure in Tcl and by "globbing" in C-shell, that is "*" stands for any sequence of zero or more characters and "?" stands for any single character. If the expression can be expanded into a set of files, the timestamp of the entity is the timestamp of the most recent file. If the set of files corresponding to the expression is empty, the entity is considered non-existing and its timestamp is 0.

This database is useful to represent the situation where a job depends on a directory and on all the files in the directory.

JOBSTATUS

This database represents the status of a job. The name of the place has the form *TYPE/JOBID* where TYPE is one of DONE, FAIL, or SUCCESS and JOBID is the id of a job. The timestamp of the place is a fixed timestamp in the past and never changes. What changes is the status of the place depending on the status of the job it is attached to. For more information on this database, please refer to [The JOBSTATUS Database](#).

LINK

This database behaves like the FILE database, except that the entities are symbolic links. An easy way to add links to the flow is to use the option `-links` in the calls to VovInput and VovOutput (either encapsulation or instrumentation)

```
...
VovInput -db LINK nameOfLink
VovInput -links someInputFile
VovOutput -links someOutputFile
...
```

PHANTOM

This database behaves like the FILE database, except that the timestamp of a missing file is, by convention, a time in the distant past (by Computer Science standards), i.e. sometime in the year 1970. This database is useful to represent situations where a job is sensitive to the fact that a file may exist or not, for example when a tool uses a search path to locate files.

VOVSETS

The entities in this database are sets of nodes. The timestamp of the entity is the timestamp of the set creation. Each set in the trace has a name and a timestamp.

ZIP

The entities in this database are members in an ZIP archive. The name of the entity has the form ARCHIVENAME (MEMBERNAME) .

The timestamp of the entity is the timestamp of the member inside the archive. In encapsulation scripts, be careful to escape the parentheses appropriately as in the following example:

```
...  
VovOutput -db ZIP -ignore_timestamp $archiveName\($memberName\  
...
```

The LINK Database

To establish dependencies with symbolic links, the LINK database must be used.

This section assumes that are familiar with the "symbolic link" in a UNIX file system. For information about symbolic links, you can use the UNIX command `man ln`.

For example, the tool "ln" can be used to create a link. In FDL, write:

```
# Create a link called 'bar' for to a file called 'foo'  
# bar → foo  
R "unix"  
T vw ln -s foo bar  
O -db LINK bar
```

Notice that the file `foo` in the example above does not even need to exist, and therefore it is not considered an input to "ln". The link `bar → foo` is created whether `foo` exists or not. The job `ln -s ...` can be run at any time, even before `foo` exists, but not before the jobs that use the symbolic link `bar`.

The timestamp of the link is not the same as the timestamp of the file. You can have an old link that points to a young file. Recreating the link will normally change the timestamp of the link.

It is possible to create automatic dependencies for all links used in path expansion. If you are using the VIL tools, you can use the option `-links` at runtime:

```
### ---- in the middle of a script ....  
  
### All symbolic links used to expand foobar and abc will be declared  
### as inputs to the current job.  
VovOutput -links foobar || exit 1  
  
VovInput -links abc || exit 1
```

All symbolic links are declared as input dependencies. (The only tool that has a LINK as output is "ln" or some other application that call the `symlink()` system call.)

Alternatively, you can set the environment variable `VOV_VW_TRACK_LINKS` before you invoke the tool to track all symbolic links.

```
setenv VOV_VW_TRACK_LINKS 1  
vw cp aa bb
```

Examples

In the following examples, the following structure are assumed (links shown in **bold**):

```
/project/ivy/releases/current      -> milestoneA  
/project/ivy/releases/milestoneA  -> serials/00014  
/project/ivy/releases/serials/0014/file.txt
```

Focus on runtime declarations using the VIL-Tools VovInput and VovOutput. Assume you are in directory /project/ivy/releases:

- ```
VovInput current/file.txt
VovInput -db FILE current/file.txt
```

These commands are equivalent, because FILE is the default database. They both create an input dependency with FILE serials/0014/file.txt, because the links "current" and "milestoneA" are both traversed.

- ```
VovInput -db LINK -quote current
```

This creates an input dependency for the symbolic link `current` → `milestoneA`. The path "current" is not expanded because it is quoted by the option -quote.

- ```
VovInput -links current/file.txt
```

This command creates three dependencies: one for FILE serials/0014/file.txt and two for the links that have been traversed. This is equivalent to these three lines:

```
VovInput -db FILE /project/ivy/releases/serials/0014/file.txt
VovInput -db LINK -quote /project/ivy/releases/serials/current
VovInput -db LINK -quote /project/ivy/releases/serials/milestoneA
```

- ```
VovInput -db LINK  current
```

Without the -quote option, this creates an input dependency for the directory serials/0014, but in the "LINK" database, which is actually a silly thing to do. Behaviorally, this is not much different from having a dependency on a "FILE" serials/0014.

- ```
VovInput -links -db LINK current
```

This is the same as above, but in addition we also have dependencies on the two links that have been traversed.

- ```
VovInput -links -db FILE  current
```

This is better than above, because the expanded path "current" is actually a directory, and not a link.

Option -links in FDL

The option -links is available also in the procedures I and O in FDL, but they have no effect. The symbolic links are only added as inputs when the jobs are actually run.

```
## This is legitimate FDL.  
T vw cp aa bb
```

```
O -links bb  
I -links aa
```

```
# More useful is to use -links in a capsule-on-the-fly  
T vw ./myjob aa bb  
CAPSULE {  
    I -links aa  
    O -links bb  
}
```

Debugging symlinks

If you want to get more verbose information about handling of symlinks, you need to add the following statement in the `equiv.tcl` file:

```
# This goes in the equiv.tcl file.  
# It activates debugging for both the equivalence subsystem  
# and the symlink subsystem  
vtk_equivalences_debug 1
```

This change will have an effect when you run another job. Remember to revert the change after you are done debugging.

Links into the DesignSync Caches


Some links in particular need not be traversed. One example of this are the links into the DesignSync caches. To define which caches to ignore, use the variable `VOV_SYNC_CACHE_DIR`, which is a colon-separated list of paths.

```
##  
## Avoid expanding links into the DesignSync caches.  
## This is normally set in the setup.tcl file.  
W#  
setenv VOV_SYNC_CACHE_DIR /some/location/sync1/cache:/other/location/sync33/cache
```

The JOBSTATUS Database

Places in the JOBSTATUS database represent the status of jobs.

The name of the place is of the form *TYPE/JOBID*: TYPE represents DONE FAIL or SUCCESS; JOBID is typically the ID of the job containing the place.

 **Note:** JOBID could be any unique string.

This database does not require any operation on any filesystems. It is used in Accelerator to represent jobs that have no output log. It is also used to represent jobs whose execution depends on the success or failure of a job.

To attach a JOBSTATUS place to a job, use the following FDL code:

```
# Fragment of FDL  
set jobid1 [J vw cp aa bb]
```

```
O -db JOBSTATUS -sticky -quote "DONE/$jobid1"

# Run this job after jobid1, whether it passes or fails.
set jobid2 [J vw cp bb cc]
I -db JOBSTATUS -sticky -quote "DONE/$jobid1"
```

Type	Behavior
SUCCESS	This is the simplest case. The place becomes VALID when the containing job also becomes VALID. If there was a barrier on the place, the barrier is removed.
FAIL	The place becomes VALID only when the containing job becomes FAILED. At the same time, a barrier is added to the place in order to preserve the trace consistency, since it is not legal for an output of a FAILED job to be VALID unless there is a barrier on the output place.
DONE	This is a combination of the two cases above. The place becomes VALID only when the containing job completes, when it becomes either VALID or FAILED. At the same time, a barrier is added or removed depending on the status of the job.

Define Equivalences for File Names

There are multiple methods to define the equivalences used to compute the canonical names of files and directories.

For example:

- Instruct the server to parse the `equiv.tcl` file and provide entries to clients. This is the default behavior. Note that for this case, equivalences that reference an environment variable should not resolve the variable in this file, in environments that will have both UNIX and Windows clients. Instead, they will need to be resolved by the client upon receipt. This is done by enclosing the equivalence value inside curly braces and referring the environment variable as `$VARNAME` as opposed to the Tcl format of `$env(VARNAME)`.
- 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 the `equiv.tcl` file for entries, and read/write access to the `equiv.caches` directory so the entries can be written to a host-based cache file for future use. In this mode, environment variables may be resolved in this file, but the behavior will be the same as not allowing them to be resolved. To resolve them in this file, the equivalence value should not be wrapped with curly braces and the environment variable should be referred to in the Tcl format of `$env(VARNAME)`. This method is enabled by setting the `VOVEQUIV_CACHE_FILE` environment variable to "legacy".
- Instruct clients to read a specific cache file only. 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. In this mode, environment variables may

be resolved in this file, but the behavior will be the same as not allowing them to be resolved. To resolve them in this file, the equivalence value should not be wrapped with curly braces and the environment variable should be referred to in the Tcl format of `$env(VARNAME)`. This method is enabled by setting the `VOVEQUIV_CACHE_FILE` environment variable to a valid equivalence cache file path.

Equivalence File

The equivalence file (`equiv.tcl`) defines the rules to generate logical names. This file is used by all clients as well as by the server. This file is a Tcl script. The fundamental procedure used in this script is `vtk_equivalence`.

The procedure `vtk_equivalence` has the following purposes:

- The main purpose is to define an equivalence between a logical name and a physical path, as in:

```
vtk_equivalence TOP /export/projects/cpu  
vtk_equivalence TOP p:/cpu
```



Note: The physical path need not be canonical. The definition is silently ignored if the physical path does not exist.

- The secondary purpose is to control the case sensitivity for file names, using the options `-nocase` or `-case`. With `-nocase`, all names are canonicalized to lowercase, which is useful when the vovserver is running on a Windows NT machine.
- The third purpose is to control whether the AFS paths should be supported. If the `-afs` option is used, then all paths of the type `/.automount/hostname1/root/aaa` become `/net/hostname1/aaaa`

The procedure `vtk_equivalence` also has side effects:

- The environment variable corresponding to the logical name is set, if it does not exist already (that is, the variable `$env(TOP)`).
- The Tcl global variable corresponding to the logical name is set to the value of the environment variable (that is, the variable `$TOP`).

Example Uses of `vtk_equivalence`

See the following example:

```
# -- HOME should not be used in multi-user projects, because it has a  
# -- different value for each user. Use it only in single-user projects.  
  
vtk_equivalence HOME $env(HOME)  
  
# -- VOVDIR is always defined and these equivalences are always useful.  
  
vtk_equivalence VOVDIR $VOVDIR  
vtk_equivalence VOVDIR $VOVDIR/./common  
  
# -- Data directories.  
  
vtk_equivalence TOP /export/projects/cpu; # This is for Unix  
vtk_equivalence TOP p:/cpu                ; # This is for Windows  
  
# Uncomment this if you need AFS paths.
```

```
# vtk_equivalence -afs
```

For another example of equivalence file, see the default file for the "generic" project type \$VOVDIR/local/ProjectTypes/generic/equiv.tcl.

Define Host-specific Overrides for the Server-side Equivalence Cache

The server-side equivalence cache can be accessed via the VTK Tcl API using `vtk_equivalence_get_cache` `OPTION`. When passing a host name in for `OPTION`, the equivalences for that host will be returned. When passing an empty string in for `OPTION`, the list of host names that have entries is returned. By default, a special host name of `"_default_"` is used for the server-side cache that applies to all clients.

The server-side equivalence cache can be set with via the VTK Tcl API using `vtk_equivalence_set_cache` `HOSTNAME` `VALUES`, where `HOSTNAME` is the name of a host or the `"_default_"`, and `VALUES` is a Tcl list with an even number of elements in the form `LOGICAL_NAME` `PHYSICAL_PATH`.

```
vtk_equivalence_set_cache lin0201 "HOMES /homes VOVDIR /tmp_mnt/tools/rtda/current/"
```

The equivalences can also be viewed and managed via the web UI on the Equivalences page.

Equivalence Cache

To avoid executing the `equiv.tcl` script over and over for each tool invocation, the results of the evaluation are stored in the subdirectory `equiv.caches` in the server configuration directory, one for each host. The caches are recomputed automatically when the equivalence file changes.

To force a refresh of the caches, use one of the following methods:

- Restart all taskers using

```
% vovtaskermgr restart
```

- Use the option `-r` in `vovequiv`, to refresh the cache on the current host:

```
% vovequiv -r
```

- Use the option `-rs` in `vovequiv`, to refresh the cache and show the result:

```
% vovequiv -rs
Directory of equiv files: .
Executable equiv file:    /Users/john/projects/mac81/vovadmin/mac81.swd/equiv.tcl
Cached    equiv file:    /Users/john/projects/mac81/vovadmin/mac81.swd/
equiv.caches/mac05
* VOVDIR      -> /Users/john/rtda/2015.09/mac05x
VOVDIR      -> /Users/john/rtda/2015.09/common
* BUILD_TOP  -> /Users/john/projects/mac81
* HOME       -> /Users/john
```

Use vovequiv to Check Equivalences

To test the equivalence file, use the command `vovequiv`. The option `-s` shows all the equivalences valid on the current host plus those computed automatically:

```
% vovequiv -s
Directory of equiv files: ${JOHN}/vov
Executable equiv file:   ${JOHN}/vov/italia.equiv.tcl
Cached equiv file:      ${JOHN}/vov/italia.equiv.caches/saturn
(Hits    0) VOVDIR      -> /remote/proj2/evolve_cdrom/evcdrom/common
* (Hits  0) VOVDIR      -> /remote/proj2/evolve_cdrom/evcdrom/sun5
(Hits    0) SUNW        -> /export/home/opt/SUNWspro
* (Hits  0) TOP          -> /home/john/Italia
* (Hits  0) VENDORS     -> /remote/vendors
(Hits    0) VENDORS     -> /rtda/vendors
* (Hits  0) SUNW        -> /opt/SUNWspro
* (Hits  0) JOHN        -> /home/john
```

To test the effect of equivalences on a specific file, use the option `-p`. For example, in your home directory you can try:

```
% cd ~
% vovequiv -p file
${HOME}/file
```

Notice how FlowTracer correctly handles dots and symbolic links:

```
% vovequiv -p ~/./file
${HOME}/file
% ln -s file foobar% vovequiv -p foobar
${HOME}/file
```

Exclude Files From the Graph

The power of FlowTracer is its ability to capture all inputs and outputs for each job. When determined as more efficient, such as ignoring details that are considered as unnecessary or of little importance, selected files can be excluded from the dependency graph.

Files are sometimes excluded from a dependency graph when a file exists on one machine but not another. For example, suppose you are running `vovserver` on a 32-bit machine (`lnx32`) but you are compiling a C-file on a 64-bit machine (`lnx64`). The compilation probably requires the file `/lib64/libgcc_s.so.1` which exists on `lnx64` but not on `lnx32`. If the compiler declares `/lib64/libgcc_s.so.1` as input, the `vovserver` (running on `lnx32`) will not find such file and therefore declare the compilation as failed. Having such a file in the dependencies is not really useful and the exclusion mechanism described in this section is an easy way to avoid these dependencies.

The exclusion rules are defined in the `exclude.tcl` file. There are two types of rules relating to a file's name that will lead to a file being excluded from the graph:

- Based on a regular expression.


- Based on a prefix: these rules are a special case of regular expressions but they execute faster,

Both rules are defined with the Tcl procedure `vtk_exclude_rule`. For example, to exclude all files in `/usr/tmp`, a prefix rule can be defined. Example:

```
vtk_exclude_rule -prefix /usr/tmp
```

To exclude all files that end with the `.tmp` suffix, a rule can be based on a regular expression. Example:

```
vtk_exclude_rule -regexp {\.tmp$}
```


 **Note:** The braces are necessary to protect the regular expression from the undesired evaluation of special characters such as `$` and `[]`.

While parsing the `exclude.tcl` file, the global variables `$argv0` and `$argv` are set to the command line of the tool, including the wrapper. This makes it possible to create different exclusion rules depending, for example, on the tool name. Additionally, the variables `EXCLUDE_TOOL`, `EXCLUDE_JOBNAME`, and `EXCLUDE_JOBCLASS` are available.

```
if { $EXCLUDE_TOOL == "dc_shell" } {  
    vtk_exclude_rule -regexp {/command.log}  
}
```

The exclude file is used by the tools and it is ignored by the server. Any change to the exclude file is effective immediately for all the tools that are executed after the change. The change, however, has no retroactive effect. Files to be excluded that are already in the graph must be forgotten explicitly with other commands.

Procedure	Arguments	Description
<code>vtk_exclude_rule</code>	<code>[-prefix </code> <code>-regexp </code> <code>-clear </code> <code>-tool TOOLNAME_RX </code> <code>-jobname JOBNAME_RX </code> <code>-jobclass JOBCLASS_RX]</code> <code>string</code>	<p>Define an exclusion rule. This procedure is used in the <code>exclude.tcl</code> file and can also be used in the encapsulation files. Examples:</p> <ul style="list-style-type: none"> Exclude all files that end begin with <code>/usr/tmp</code> <pre>vtk_exclude_rule - prefix /usr/tmp vtk_exclude_rule - tool dc_shell -regexp command.log vtk_exclude_rule - jobclass synth -regexp {.vvv\$}</pre> Exclude all files that end with a tilde. <pre>vtk_exclude_rule {~\$}</pre>

Procedure	Arguments	Description
		<ul style="list-style-type: none"> Reset the exclude rules. <div>  Note: This procedure is rarely used. </div> <div> <pre>vtk_exclude_rule -clear</pre> </div> <ul style="list-style-type: none"> Exclude the <code>command.log</code> file, but only if the tool is <code>dc_shell</code>. <div> <pre>vtk_exclude_rule -tool dc_shell -regex command.log</pre> </div> <ul style="list-style-type: none"> Exclude files ending in <code>.vvv</code> if the job is running in the job class "synth". <div> <pre>vtk_exclude_rule -jobclass synth -regex {\.vvv\$}</pre> </div>
<code>vtk_path_exclude</code>	<code>path</code>	Test whether a path is excluded by the rules that have been defined. This procedure is normally not used in the <code>exclude.tcl</code> file.

Examples of vtk_exclude_rule

For an example of exclude file, see the default file for the "generic" project type at `$VOVDIR/local/ProjectTypes/generic/exclude.tcl`.

Another example:

```
# The files that start with these prefixes will not be added to the graph.
vtk_exclude_rule -prefix ${VOVDIR}
vtk_exclude_rule -prefix /dev/
vtk_exclude_rule -prefix /devices/
vtk_exclude_rule -prefix /etc/
vtk_exclude_rule -prefix /proc/
vtk_exclude_rule -prefix /lib/
vtk_exclude_rule -prefix /opt/CC
vtk_exclude_rule -prefix /tmp/
vtk_exclude_rule -prefix /tmp_mnt/usr/
vtk_exclude_rule -prefix /usr/
vtk_exclude_rule -prefix /var/
vtk_exclude_rule -prefix c:/
```

```
vtk_exclude_rule -regex {Dependency.state$}  
vtk_exclude_rule -regex {/gcc-lib/}  
vtk_exclude_rule -prefix /lib64;    ### Added for multi-arch compilations.  
  
# -- Exclude the NT executables from the current installation.  
vtk_exclude_rule -regex {VOVDIR.*/[a-z]+\..exe$}  
  
# -- Tool dependent exclusion rule.  
# -- (the first arg is the wrapper, the second the tool)  
# -- Could also use the variable EXCLUDE_TOOL  
switch -- [lindex $argv 1] {  
    "toolx" {  
        vtk_exclude_rule -regex /TOOLXCACHE/  
    }  
}
```

Debugging the Exclusion Mechanism

Use the environment variable VOV_STRICT_TRACING to disable the exclusion mechanism in your shell.

Use the environment variable VOV_DEBUG_FLAGS to force the exclusion routines to print out debugging messages:

1.

```
% setenv VOV_DEBUG_FLAGS 128
```
2. Run the tools.

Additional Files with Exclusion Rules

The variable VOV_EXCLUDE_FILES can be used to list a number of additional files that contain exclusion rules. Examples:

```
% setenv VOV_EXCLUDE_FILES $VOVDIR/local/exclude/exclude.cdn.tcl  
% setenv VOV_EXCLUDE_FILES $VOVDIR/local/exclude/exclude.cdn.tcl:$VOVDIR/local/  
exclude/exclude.snps.tcl
```

Change the exclude.tcl File

The exclude file is used by the tools and not by the server; when an exclude file is changed, no immediate action will occur. The files that are excluded will remain in the trace if the files are already present. However, as tools are executed, the dependencies with the excluded files will be dropped.

In the following example, all dependencies with files in the directory /home/tools/bin are being excluded from the trace.

1. Add to the exclude.tcl file the line:

```
vtk_exclude_rule -prefix /home/tools/bin
```

2. Create a set of all the files to be excluded that are already in the trace:

```
% vovset create Tmp:exclude "isfile name~/home/tools/bin"
```

3. Forget all those files:

```
% vovforget -elem Tmp:exclude
```

4. Forget the temporary set:

```
% vovset forget Tmp:exclude
```

Automatic Zipping and Unzipping Files

FlowTracer can automatically compress (zip) and uncompress (unzip) files in the flow using the utilities `gzip` and `gunzip`. This service is provided for all places that satisfy the following conditions:

- In the database "FILE"
- Have the "zippable" flag set

Rule 1: Compression

A zippable file is automatically compressed when all the jobs that require compression have completed successfully. The check to see if compression is required is performed about every 30 seconds.

Rule 2: Decompression

A zippable file is automatically uncompressed if any of the jobs that requires it is queued and ready to be executed. Jobs will not be executed with compressed inputs.

If a file is compressable, it can be in either the normal or the compressed state. For a file called 'X', if the file does not exist but 'X.gz' does, then the file is considered compressed.

Directing vovzip Jobs to Certain Hosts

The work to compress/uncompress a file is performed by a system job that is executed by the owner of the project. These jobs are scheduled to be executed by the appropriate vovtaskers.

To direct the `vovzip` jobs to particular hosts, create a tool map in the project's `resources.tcl` file, where the right-hand side of the map is the necessary resource.

For example, the following statement could be used:

```
vtk_resourcemap_set Tool:vovzip UNLIMITED vovzip_host
```

The vovtasker resource `vovzip_host` can be placed on the desired hosts in the project's `taskers.tcl` file. The number of concurrent `vovzip` jobs can be limited: instead of `UNLIMITED`, insert an integer such as 10.

Defining Zippable Files

In a flow description, the procedure `z` can be used to define the files that are zippable. Example:

```
J vw cp aa bb  
J vw cp bb cc  
Z bb cc
```

Another method is using the command `vovset` with the subcommand `zippable`.

Examples:

```
% vovset zippable SETNAME 1
% vovset zippable SETNAME 0
```

Using Tcl, `vtk_place_get` and `vtk_place_set` can be used to test and set the zippable flag.

```
set id [ vtk_place_find "FILE" "aa"]
vtk_place_get $id info
puts "The current value of the zippable flag is $info(zippable) "
puts "The current value of the zipped flag is $info(zipped) "

set info(zippable) 1
vtk_place_set $id info
```

Recommendations for Zippable Flags

Using the zippable flag is recommended for relatively large files, such as greater than 100kB. A tradeoff needs to be evaluated: the space consumed and flow complexity and additional CPU/IO time, versus the degree of the compressibility of the files. For example, JPEG and PDF files are already highly compressed; zipping these files may produce little or not benefit. Test vector files, however often compresses to 3-8% of their native size.

vovzip Jobs and Log Files

The information in this section may be helpful for troubleshooting zippable files. The jobs that compress and expand the zippable files:

- Are created automatically by the vovserver.
- Use the tool `vovzip`, supplied by FlowTracer.
- Are retraced in "FAST" mode.
- Their log file goes into the directory `projectName.swd/vovzipdir`.
- The name of the log file is of the form `FILEID.log`, where `FILEID` is the `VovId` of the zippable file.

Taskers

A tasker is a VOV client that provides computing resources, specifically CPU cycles, to the vovserver.

There are two types of taskers:

- **Direct taskers:** agents that offer for computation all the resources of the machine on which they are running
- **Indirect taskers:** agents that interface between a VOV project and a scheduler such as Accelerator.

A project can have a mix of direct and indirect taskers. Normally, Accelerator and Monitor use only direct taskers, while FlowTracer projects often interface to schedulers using one or more indirect taskers.

The list of taskers connected to a project is described in the file `taskers.tcl`, and the main utility to start and stop taskers is `vovtaskermgr`. Additional configuration can be specified with the `taskerClass.table` file.

Types of vovtasker Binaries

The main tasker client is called `vovtasker` but there are other variations of it:

- `vovtasker` can run jobs for more than one user; the success depends on file permissions.
- `vovtaskerroot` has the ability of switching user identity.



Note: Accelerator is the only Altair Accelerator product that needs `vovtaskerroot`.

- `vovtasker.exe` has the ability of impersonating users on Windows, subject to a set of rules explained in *Vov Windows Impersonation*.
- `vovagent` is a temporary `vovtasker` that terminates upon a set of timeouts, and is used mostly in conjunction with LSF or SGE.
- `ftlm_agent` is a "thin-client" version of a `vovtasker` and can be used by Monitor to start and stop license daemons on remote machines.

vovtasker States

The `vovtaskers` will change states based on workload and operating environment. Generally, the tasker state will accompany the tasker name when displayed in the various user interfaces (CLI, GUI, WUI).

The possible states are as follows:

- **BLACKHOLE:** temporarily paused due to a burst of job failures and cannot accept jobs
- **BUSY:** busy with internal operations
- **DEAD:** has disconnected and cannot accept jobs
- **DONE:** exiting after completing current jobs
- **FULL:** full and cannot accept more jobs
- **OVRLD:** overloaded and cannot accept jobs
- **NOLIC:** unlicensed and cannot accept jobs

- **NOSLOT:** configured to not accept job
- **OK:** idle and ready for jobs
- **PAUSED:** paused and cannot accept jobs
- **READY:** idle and ready for jobs
- **REQUESTED:** has been requested to start
- **SICK:** sick and possibly disconnected
- **SUSP:** suspended and cannot accept jobs
- **WARN:** in a warning state and should be checked
- **WRKNG:** working and can accept more job

The taskers.tcl File

The `taskers.tcl` file describes the taskers for a project.

The `taskers.tcl` file is a Tcl script based on the procedure `vtk_tasker_define`. The synopsis for this procedure:

```
vtk_tasker_define hostname [options]
```

The two following examples both declare three taskers on the hosts apple orange and pear:

```
# Fragment of taskers.tcl file
vtk_tasker_define apple
vtk_tasker_define orange
vtk_tasker_define pear
```

```
# Fragment of taskers.tcl file
foreach host {apple orange pear} {
    vtk_tasker_define $host
}
```

The following procedure supports many [options](#) to define the characteristics of the tasker. The options include `"-resources <string>"` to set the resource list offered by a tasker and `-CPUS n` to define the number of CPUs in a machine. In the following example, the tasker on apple is set up to offer the resource `"big_memory"`:

```
# Fragment of tasker file
vtk_tasker_define apple -resources "@STD@ big_memory" -CPUS 2
vtk_tasker_define orange -resources "@STD@ big_memory" -CPUS 2
vtk_tasker_define pear -resources "@STD@ big_memory" -CPUS 4
```

The default value for all options can be changed with the following procedure `vtk_tasker_set_defaults`, as shown below:

```
# Fragment of taskers.tcl file
vtk_tasker_set_default -resources "@STD@ big_memory" -CPUS 2
vtk_tasker_define apple
vtk_tasker_define orange
```

```
vtk_tasker_define -CPUS 4
```

vtk_tasker_define

There are many options that can be used with `vtk_tasker_define` and `vtk_tasker_set_default`.

Option	Argument	Description
-capabilities	string	The capabilities that the tasker has. This results in license checkout attempts for the specified capabilities. Possible values: FULL, NC, PROCINFO, NETINFO, EXEC, RT. FULL includes all capabilities. NC contains EXEC, PROCINFO and NETINFO capabilities. Altair Accelerator enables runtime tracing for FlowTracer. Default is FULL.
-capacity	int	The number of concurrent jobs (job slots) that the vovtasker can handle.
-cpus	int	The number of CPUs in the machine, without affecting maxload and capacity. On most platforms, the number of CPUs is computed automatically.
-CPUS	int	Convenience option, equivalent to setting -cpus, -maxload and -capacity at the same time. If N is the number of CPUs, this options sets -cpus to N , -maxload to $N+0.5$ and -capacity to N .
-coeff	double	The tasker coefficient. It is used as a divisor in computing the effective power of a vovtasker, e.g. a coefficient of 2.0 reduces the power by half.
-executable	string	The executable to use (default is vovtasker). For Accelerator, the default is vovtaskerroot.
-expiredate	string	Specifies the date and time after which the definition of this tasker is expired, and it cannot be started with vovtaskermgr command. The format of this parameter is year_month_day_hour_min_sec. Example: 2018_12_31_23_59_00
-failover		Passing this option will set the tasker's capacity to 0, which prevents the tasker from accepting jobs and pulling a license. This also acts as a flag to

Option	Argument	Description
		perform some failover configuration testing, such as checking <code>servercandidates.tcl</code> to make sure the tasker host is in the list, triggering a check to make sure the host has at least as many file descriptors as <code>vovserver</code> so it can operate at full capacity in the event of failover, and checking that the <code>server_election</code> directory is empty.
-host	hostname	The name to be used to connect to the vovserver (e.g. "localhost")
-indirect	file	Execute the jobs indirectly, using the Tasker* procedures described in the given file. This is used in indirect taskers.
-maxcapacity	int	The capacity of taskers can increase dynamically as a side effect of having suspended jobs. This limits the maximum capacity. The default value is twice the capacity.
-maxjobs	int	The maximum number of jobs a tasker can execute. Taskers will become suspended when the max is reached, and will exit once the last job is finished. Default: 0 (unlimited).
-maxidle	timespec	The maximum amount of time a tasker can be idle (having no jobs). Default: - (unlimited).
-maxlife	timespec	The maximum amount of time a tasker is allowed to run. Default: - (unlimited).
-maxload	double	The maximum load for the vovtasker. Above this, its power becomes zero, and the vovtasker does not accept new jobs until the load declines below this value. This helps avoid overloaded machines.
-message	string	Message to set on the vovtasker at startup. Should be brief.
-mindisk	number	Minimum disk space, in MB (for example, 100) or in percentage (0%-99%, for example, 10%) , on <code>/usr/tmp</code> below which the vovtasker will automatically be suspended.
-name	string	Name of the vovtasker. The default is the leaf name of the machine on which the vovtasker runs. May not contain the '.' (dot) character.


Option	Argument	Description
-nice	int	Run the tasker with niceness (reduced OS priority), for UNIX vovtasker only, ignored on Windows.
-power	double	The raw power to be used for this vovtasker. The default for this is 0.0, which implies that the raw power is computed automatically upon startup. You can use this to make machines know to be identically- provisioned to have the same power.
-repeat	int	Number of identical vovtasker on a host (obsolete).
-reserve	reserve expression	Reserve the vovtasker upon startup. The argument is a reservation expression. Example 1: "--reserve /john/3w" This means create a reservation for user john with a duration of three weeks at vovtasker startup. Example 2: "--reserve memregr//4h" This means create a reservation for group 'memregr' with a duration of 4h at vovtasker startup.
-resources	string	The tasker resources. This could be a list of literals like "bighost maingroup" or contain symbolic values like "@RAM@ @CPUS@". You must restart vovtaskers after changing values specified here. For a method that does not require restart, read about The taskerClass.table file.
-rshcmd	string	The command used to start a remote shell on the tasker machine. This is rsh by default, but it could be set, for example, to ssh. The known values for this option are: <ul style="list-style-type: none"> • rsh, the default value • ssh, the typical value • vovtsd
-serverdir	dir	Explicit path to the server directory for the tasker.
-taskergroup	group	Define the taskergroup in <code>taskers.tbl</code> . Often this is used to group similarly-provisioned machines.
-update	timespec	The update cycle time (heart beat) of the vovtasker. The default value is 60s. You can use

Option	Argument	Description
		shorter values to cause resources to be updated more frequently when using resource procedures, being mindful of the CPU load this brings to the vovserver.
-vovdir	dir	Explicit path to the VOV installation for the tasker.
-vovtsdport	port	Port number to be used when connecting to the VOV tasker service daemon, vovtsd.

vovtaskermgr

The main way to start, configure, and stop the taskers is with the `vovtaskermgr` command. This command acts relative to the VOV-project enabled in the shell where it is issued.

The file `taskers.tcl` in the `project.swd` directory stores the configuration information used by this command.

 **Note:** Changes made to `taskers.tcl` are not automatically propagated to the running vovtaskers. To do this, use the `update` subcommand.

A vovtasker listed in the `taskers.tcl` file may be running or stopped. The `show` subcommand gives information on the running vovtaskers currently connected to the vovserver. The `list` subcommand gives the names of all the vovtaskers defined in vovtaskers, whether running or stopped.

```
vovtaskermgr: Usage Message

USAGE:
  vovtaskermgr <SUBCOMMAND> [options] [taskerList]

  SUBCOMMAND is case-insensitive.

  The taskerList consists of tasker names or tasker ids.

SUBCOMMAND is one of:
  LIST          -- List all hosts named in the taskers.tcl file.
  RESTART       -- Same as STOP followed by START.
  REFRESH       -- Refresh cached environments and equivalences.
                  The default behavior is for taskers to obtain the
                  equivalences from the server. If changes are made to the
                  equiv.tcl file, the server will need to be instructed to
                  reread the file using the "vovproject reread" command
                  prior to requesting a tasker refresh.
                  If VOVEQUIV_CACHE_FILE is set to "legacy", a host-based
                  equivalence cache file will be created and updated in
                  the SWD/equiv.caches directory. If VOVEQUIV_CACHE_FILE
                  is set to a file path, the specified file will be used
                  instead.
```

```

SHOW          -- Show info about connected or down taskers.
PRINTSTATUS   -- Tell taskers to print their status in their log file.
START         -- Start configured taskers.  If a list of hosts is
               given, start taskers only on those hosts.  Otherwise,
               start all configured taskers that are not running.

UPDATE        -- Update configuration of running taskers.
RESERVE       -- Reserve specified taskers.
RESERVESHOW   -- Show current tasker reservations.
CONFIGURE     -- Reconfigure the specified taskers on-the-fly.
               Changes only persist until the tasker is stopped.
STOP          -- Stop taskers; let jobs finish, unless -force is given.
CANCELSHUTDOWN -- Revert stopped but still running taskers to normal
               so they continue running and accept new jobs.
ROTATELOG     -- Recreate new log files for specified taskers
               if log files are missing, create tasker log directories
               if needed, and have no impact on tasker startup logs.
CLOSE [MSG]   -- Close taskers from accepting jobs.  Closed taskers will
               start and run, but will do so in a suspended state,
               displaying the closure message, until opened by the
               administrator.  The default closure message is
               'Closed by administrator'.
OPEN [MSG]    -- Open taskers to accept jobs.  The accompanying message
               will be displayed on running taskers until another
               message is generated during the course of normal
               operation.  Taskers that are not running will not display
               the message after starting.  The default opening message
               is an empty string.

```

Global Options are:

```

-l           -- Use longer format with LIST (may be repeated).
-v           -- Increase verbosity of messages.
-cfgfile     -- Specify path to tasker config file, relative to SWD.
               Default: taskers.tcl
-failover    -- Restrict operation to dedicated failover taskers only.

```

Options for SHOW are:

```

-nameonly    -- Show only the names of the connected taskers.
-nameid      -- Show only the names and ids of the connected taskers.
-resourceonly -- Show only the resources of the connected taskers.
-down        -- Show names of configured taskers that are down.
-license     -- Show licensed capabilities of connected taskers.
-taskergroups -- Show tasker group for each connected tasker.

```

Options for START and RESTART are:

```

-server      -- Start the taskers by rsh/ssh from the vovserver host.
               By default, the taskers are started
               by the host that executes this script.
-random      -- Start taskers in random order.
               This is useful to start a large pool of tasker,
               by running multiple concurrent commands like:
               % vovtaskermgr start -random &
               % vovtaskermgr start -random &
               % vovtaskermgr start -random &
-nolog       -- Redirect tasker output to /dev/null.
               Useful to avoid huge log files in /usr/tmp
-confirmafter <TIMESPEC>
               -- Wait for the given time specification after the last start
               request for the list of taskers being started, then print
               whether each tasker has successfully started and connected
               to the vovserver.  Only taskers in the READY, WRKNG, FULL, or
               OVRD state will be considered as running.

```

Options for RESERVE are:

```

-user        -- Reserve the tasker(s) for given list of users

```

```

        (comma separated list)
-usergroup  -- Reserve the tasker(s) for given list of user groups
              (comma separated list)
-group      -- Reserve the tasker(s) for given list of FairShare groups
              (comma separated list)
-jobclass   -- Reserve the tasker(s) for given list of jobclasses
              (comma separated list)
-jobproj    -- Reserve the tasker(s) for given list of job projects
              (comma separated list)
-osgroup    -- Reserve the tasker(s) for given list of Unix groups
              (comma separated list)
-bucketid   -- Reserve the tasker(s) for given list of queue buckets
              (comma separated list)
-id         -- Reserve the tasker(s) for given list of jobs
              (comma separated list of job ids)
-start      -- Reservation start time
-end        -- Reservation end time
-duration   -- Reservation duration (VOV timespec)
-cancel     -- Cancel the reservation on tasker(s)
-hardfill   -- Backfill the reservation with only
              autokill jobs
-softfill   -- Backfill the reservation with only
              autokill or xdur jobs

```

Options for STOP are:

```

-force      -- Stop taskers with force. BEWARE: kills running jobs.
-noconfirm  -- Do not prompt for confirmation. Default is to prompt.
-all       -- Stop all running taskers.
-sick <TIMESPEC>
              -- Stop all taskers that have been sick for at least the given
              time specification, as compared against the last time a
              heartbeat was received by the server for each sick tasker.
              All jobs running on a sick tasker being stopped will be
              marked as failed in the server, even if the job does,
              or has, completed successfully while the tasker is sick.
              It is recommended to check tasker host connectivity before
              using this function and allow for the tasker to reconnect
              and send a heartbeat in case connectivity is restored.

```

Parameters for CONFIGURE are:

```

-allowcoredump <bool>  -- Control core-dump behavior.
-autokillmethod <d|n|v> -- Control autokill method.
-capacity <CAP>[MAXCAP] -- Specify capacity and optionally the
                          max-capacity of the tasker. The capacity is
                          the maximum number of jobs that can be run by
                          tasker. The max_capacity is the maximum slots
                          a tasker can be expanded to have when jobs are
                          suspended. The default value for capacity is
                          equal to the number of CORES present. The
                          default value for max_capacity is 2*CAPACITY.
                          Use N, N/N, CORES[-+*/]N, CORES[-+*/]N/N,
                          N/CORES[-+*/]N, CORES[-+*/]N/CORES[-+*/]N to
                          make adjustments from the default.
                          Examples: 4, 4/8, CORES-2, CORES*0.8,
                                   CORES+0/20, CORES+2/CORES*2
-cpus <N>              -- Number of CPUs in this machine.
-debugcontainers <bool> -- Enable debug logging of container activity.
-debugjobcontrol <bool> -- Enable debug logging of job control activity.
-debugmultienv <bool>  -- Enable debug logging of environment switching.
-debugnuma <bool>      -- Enable debug logging of NUMA activity.
-debugusageinfo <bool> -- Enable debug logging of memory usage analysis.
-maxload <MAXLOAD>     -- Maximum load above which new jobs are refused.
                          The default value for max_load is

```

```

CAPACITY+0.5.
Use 0 or less than 0 to specify default value.
Use N or CAPACITY[-+*/]N to make adjustments
from the default.
Examples: 12.0, CAPACITY+2, CAPACITY*2
-maxwaitnostart <N>      -- How long to wait for a job to start.
-maxwaittoreconnect <N> -- How long to wait before reconnect.
-message <string>       -- Set vovtasker message.
-numabindtonode <bool>   -- Bind to entire NUMA node or individual cores.
                        -- Default is to bind to entire NUMA node.
-resources <string>      -- vovtasker resources.
-taskergroup <string>    -- The tasker group.
-minramfree <N>          -- Minimum amount of free RAM in MB.
-name <string>           -- Name of vovtasker.
-ramsentry <bool>        -- Activate/Deactivate RAM SENTRY.
-efftotram <N>           -- Effective total RAM in MB.
-retrychdir <N>          -- Specify number of retries for failed chdirs.
-retrychdirsleepp <N>    -- Specify the sleep interval time between
                        -- retries for failed chdirs.
-retrychdirbackoff <N>   -- Specify the factor multiplied to the sleep
                        -- interval to increase sleep interval between
                        -- retries for failed chdirs.
-liverecorder on|off     -- Enable/disable LiveRecorder debugging
                        -- capability (linux64 only).
-liverecorder.logdir <string>
                        -- Specify the directory in which the LiveRecorder
                        -- recording file should be saved. The
                        -- directory must exist. Default is "/tmp".
-liverecorder.logsize <N> --
                        -- Specify the LiveRecorder log size in MB.
                        -- Default: 256, Min: 256, Max: 65536.
-liverecorder.mode <string>
                        -- Specify the LiveRecorder mode, which is one of
                        -- the following: tasker, subtasker, both.
                        -- Note that enabling subtasker recording results
                        -- in a recording file for each job executed on
                        -- the tasker.
                        -- Default: tasker.
-rawpower                -- Specify a raw power figure for initial tasker
                        -- startup.
-mindisk                 -- Specify minimum /tmp disk in MB or
                        -- percentage (0%-99%, for example, 10%)
                        -- required for tasker startup.
-coeff                   -- Specify a scaling factor from 0.01-100.0
                        -- used to derate tasker power.
-sendenv <name>          -- Send a named environment to a tasker.
-setenv VAR=VALU E       -- Set a variable in the tasker environment.
                        -- ("VAR=VALUE" must be quoted on Windows)
-taskerheartbeat <N>     -- Specify the heartbeat for a tasker.
-unsetenv VAR             -- Unset a variable in the tasker environment.
-hardbound <bool>        -- Dispatch autokill jobs only.
-softbound <bool>        -- Dispatch autokill or xdur jobs only.

```

EXAMPLES:

```

% vovtaskermgr show
% vovtaskermgr show -nameid
% vovtaskermgr start
% vovtaskermgr start unix1
% vovtaskermgr start -random      -- Start taskers in random order.
% vovtaskermgr update
% vovtaskermgr restart
% vovtaskermgr stop              -- Stop all taskers, let running
                                -- jobs finish.

```

```
% vovtaskermgr stop -noconfirm          -- Like above, no confirmation
                                         required.
% vovtaskermgr stop -force              -- Kill running jobs now
                                         (-noconfirm implied).
% vovtaskermgr reserve -user john \\  
    -duration 3h jupiter                -- Reserve tasker jupiter for user
                                         john for 3h from now
% vovtaskermgr configure -message "shutdown 1PM" farm11 farm12
% vovtaskermgr printstatus farm11
% vovtaskermgr rotatelog                -- Recreate missing log files for
                                         all connected taskers
% vovtaskermgr rotatelog farm2 farm11   -- Recreate missing log files for
                                         tasker farm2 farm11
% vovtaskermgr configure jupiter -sendenv BASE
                                         -- send the BASE environment to
                                         tasker jupiter
% vovtaskermgr reserve -hardfill -user john \\  
    -duration 3h jupiter                -- Reserve tasker jupiter for user
                                         john for 3h from now, and backfill
with autokill jobs
% vovtaskermgr configure jupiter -softbound 1
                                         -- Reserve tasker jupiter for jobs with
autokill or xdur specified
```

Starting Many Taskers in Parallel

If you have hundreds of taskers to start, it may take some time. You can speed up the process by running multiple start script with the -random option, which is useful to start taskers in random order.

For example:

```
% vovtaskermgr start -random &
% vovtaskermgr start -random &
% vovtaskermgr start -random &
% vovtaskermgr start -random &
% vovtaskermgr start -random &
% vovtaskermgr start -random &
```

Tasker Configuration on the Fly

Many vovtasker characteristics can be changed on the fly using vovtaskermgr configure. For example, you can change the capacity of a tasker, i.e. the maximum number of jobs that the tasker can take, with:

```
% vovtaskermgr configure -capacity 8 pluto
```

Setting the capacity to zero effectively disables the tasker:

```
% vovtaskermgr configure -capacity 0 pluto
% vovtaskermgr configure -message "Temporarily disabled by John" pluto
```

Tasker Capacity

The behavior of manually overriding vovtasker cores and capacity has been improved. By default, the capacity follows the core count, but it can also be manually set via the -T option or by defining the

SLOTS/N consumable resource via the -r option, where N is a positive integer. In all cases, the capacity directly affects the number of slot licenses that will be requested.

Tasker Reservation

Below is an example of using `vovtaskermgr` to set a reservation on a tasker. In this case, you want to reserve the tasker called 'pluto' for user 'john' for 2 days.

If you wish for the vovtaskers to be reserved when they start, use the -reserve option in the `taskers.tcl` file.

```
% vovtaskermgr reserve -user john -duration 2d pluto
```

Tasker ID

Each tasker is an object in the VOV database and therefore it has its own VovId.

This ID is used in the following commands:

- `vovtaskermgr` (for command line operation)
- `vtk_job_control` to stop, suspend, resume jobs
- `vtk_tasker_reserve` to reserve a tasker.

Tasker Attributes

A tasker is characterized by several attributes. These attributes are controllable by means of the command line arguments to the vovtasker binary as well as by means of the procedure `vtk_tasker_define` in the `taskers.tcl` configuration file for a VOV project.


For cases where the vovtaskers are started by submitting the binary to a separate batch queue system, the system manager may create a copy of the vovtasker binary called `vovagent`. When invoked by this name, the binary will limit the values of some attributes based on information stored in the configuration file `$VOVDIR/local/vovagent.cfg`.

Attribute Name	vovtasker options	vtk_tasker_define option	Description
name	-a	-name	Name of the tasker. By default, it is the name of the host on which the tasker is running. The tasker name can contain alphanumeric characters, dashes and underscores. It must be less than 50 characters long. If the name ends with a "_r", it probably indicates a tasker that has

Attribute Name	vovtasker options	vtk_tasker_define option	Description
			<p>reconnected to the tasker after a server crash. These taskers are used to terminate the jobs executing at the time of the crash.</p> <p>The name of a running tasker can be changed with <code>vovtaskermgr configure -name ...</code> or using the API <code>vtk_tasker_config \$TASKERID name "newname"</code>.</p>
capability	-b	-capabilities	<p>The capabilities that the tasker has. This results in license checkout attempts for the specified capabilities. Possible values: FULL, NC, PROCINFO, NETINFO, EXEC, RT. FULL includes all capabilities. Accelerator contains PROCINFO and NETINFO capabilities. Default is FULL.</p>
capacity	-T	-capacity	<p>Maximum number of jobs that can be run by the tasker concurrently. Default is 1 slot per core detected or specified (see -C below).</p>
maxload	-M	-maxload	<p>Maximum allowed load on the tasker host (default $N+0.5$, where N is the number of cores detected). The maximum load is the point at which the host of the tasker is too busy to accept any more jobs. A machine is considered overloaded if its load average for either the last minute or the last five minutes exceeds this boundary.</p>
loadsensor	-L	-loadsensor	<p>Use an SGE style load sensors to control power of a tasker and other resources.</p>
coefficient	-c	-coeff	<p>The tasker coefficient (positive floating point number, with default 1.0). This attribute is used to adjust the raw power. A coefficient of 1.0 indicates the actual computed power of the tasker should be used. A coefficient of 4.0</p>

Attribute Name	vovtasker options	vtk_tasker_define option	Description
			indicates the actual power of the tasker should be divided by 4.
cpus	-C	-cpus	The number of CPUs in this machine. Default is the number of cores reported by the operating system. This attribute affects the computation of the actual load on the machine and by default, defines the capacity of the tasker (see -T above).
logfile	-l	-logfile	The name of the file for the tasker log. The file name is relative to the server working directory. The file name can contain also the following symbolic strings, which will be appropriately substituted: @NUMBER@ @TASKERHOST@ @SERVERHOST@ @PROJECT@.
reserve	-e	-reserve	The reservation expression for this tasker. The argument is in the format "GROUP/USER/DURATION", where the GROUP and USER fields are optional. Examples: /john/2wusers//100dregression/ john/2w
resources	-r	-resources	The tasker resources offered by this tasker. The resource management determines the type of jobs the tasker may accept.
Remote Shell Command	n/a	-rshcmd	The command used to start a remote shell tasker on the tasker machine. The default is <code>rsh</code> (or <code>remsh</code>). Other possible values are: <ul style="list-style-type: none"> <code>ssh</code> (most common value) <code>vovtsd</code> (useful for Windows taskers), which requires <code>vovtsd</code> to be running on the remote machine.
Tasker Environment	n/a	-taskerenv	A space-separated list of VAR=VALUE elements that specify additional environment variables that need to be

Attribute Name	vovtasker options	vtk_tasker_define option	Description
			defined when starting the vovtasker. This parameter is only active when the tasker is started. To change the environment in a running vovtasker you have to use the vtk_tasker_config API.
Tasker Group	n/a	-taskergroup	The group(s) the tasker is in for purpose of viewing in the browser UI or the vovmonitor. This currently has no use other than making it easier to view groups of taskers.
timeleft	N/A	N/A	The time the tasker has left before it suspends itself. Also, the maximum expected duration for a job dispatched to this tasker. This attribute is available in the context of time variant resources and is controlled only by means of the procedure vtk_tasker_set_timeleft.
transient	-i	-transient	A transient tasker is destroyed when the vovtasker client is terminated. On the other hand, a non-transient tasker persists in memory even when vovtasker is terminated. Its status will be "DOWN".
Use Vovfire	-E	-usevovfire	With this option, a direct tasker may use <code>vovfire</code> to execute jobs instead of the direct execution of the job. Since <code>vovfire</code> does the directory change and the setting of the environment, the tasker does less work. The environment caching of the tasker is disabled in this mode. This is a development option that is useful mostly on Windows.
update	-U	-update	The period used by the tasker to update its status. The argument is in seconds. The default is 60 seconds.
mindisk	-D	-mindisk	The amount of free disk on <code>/usr/tmp</code> below which the vovtasker is suspended. The default is 5MB. Many system commands and some VOV ones depend on scratch space here.

Attribute Name	vovtasker options	vtk_tasker_define option	Description
			 Note: The value can be set to 0 to turn off tasker suspension, but incorrect operation of some commands may occur.
maxidle	-z	N/A	After being idle for the given time, tasker does not accept new jobs and exits after completing active jobs. Value is a FlowTracer timespec, for example, 2m. If unspecified, idle time is unlimited.
maxlife	-Z	N/A	After the specified lifetime, tasker does not accept new jobs and exits after completing active jobs. Value is a FlowTracer timespec, for example, 2H. If unspecified, lifetime is unlimited.
maxjobs	-m	-maxjobs	The maximum number of jobs a vovtasker will execute during its lifetime. When the max is reached, vovtasker self-suspends so it stops accepting new jobs, and will exit once its last running job finishes. Default: 0 (unlimited).

Tasker Configuration Parameters

The following table shows the vovtasker parameters that can be configured. All parameters can be configured with `vtk_tasker_config` and most parameters can be configured with `vovtaskermgr configure ...`

Parameter Name	Values	Description
<code>allowcoredump</code>	boolean	Allow dumping of core on error (normally off)

Parameter Name	Values	Description
<i>capacity</i>	non-negative	Change number of slots in vovtasker
<i>coeff</i>	0.01 -- 100.0	The tasker coefficient (used to derate power)
<i>debugnuma</i>	boolean	Print debugging messages about NUMA control
<i>efftotram</i>		Effective Total RAM
<i>expirelogs</i>	timespec	Force logs of tasker to expire after specified duration. Default expiration is next midnight
<i>maxload</i>	positive double	Change max load allowed on tasker
<i>maxwaitnostart</i>	timespec	Time to wait before killing a job that does not want to start. Default is 1m.
<i>message,sys</i>	message	
<i>message,usr</i>	message	
<i>message</i>	message	
<i>mindisk</i>	integer	Minimum disk required in /tmp (in MB) to accept a job
<i>minramfree</i>		Minimum RAM that has to be free to accept a job
<i>name</i>	New tasker name	Change name of tasker
<i>printstatus</i>	1	Print status of tasker
<i>ramsentry</i>	boolean	Activate RAM Sentry mechanism
<i>rawpower</i>	int	Change raw power of tasker
<i>refresh</i>	(ignored)	Force tasker to refresh environment and clear all caches
<i>resources</i>		Change tasker resources
<i>shutdowncancel</i>	(ignored)	Cancel a shutdown order.

Parameter Name	Values	Description
<i>taskergroup</i>	name of tasker group	Change group of the tasker
<i>update</i>		Change update time for tasker (default 1m)
<i>verbose</i>	0,1,2,3,4	Change verbosity level (for debugging)
<i>waitafterjc</i>	0-30	Time to wait after a job control action
<i>waitafterjcext</i>	0-30	Time to wait after a EXT job control action

Tasker Power

Upon startup, the vovtasker computes the *raw power* of the CPU on the current host by timing a known test routine consisting of a balanced mix of integer, double and string operations. The power is inversely proportional to the time it takes to execute the test routine.

To maintain the efficiency of the vovserver for other tasks (such as scheduling), only the 2 most significant digits of the calculations are used to search for and identify the fastest tasker host.

Here is a sample output of the startup phase of a tasker, which shows the results of the power measurements:

```
% vovtasker -N
vovtasker Jan 27 17:49:03
    Copyright © 1995-2025, Altair Engineering
    Sun5/5.2 Jan  5 2000 08:28:50
    evcdrom@mercury
vovtasker Date Stamp: Thu Jan 27 17:49:04 2000
vovtasker Jan 27 17:49:04 Test 1: INTEGER OPS W= 1.00 Reps= 500 T= 16.95ms
vovtasker Jan 27 17:49:04 Test 1: DOUBLE OPS W= 1.00 Reps= 25 T= 73.64ms
vovtasker Jan 27 17:49:04 Test 1: CHAR OPS W= 0.10 Reps= 10 T= 74.12ms
vovtasker Jan 27 17:49:04 ---- Weighted time: 98.01ms
vovtasker Jan 27 17:49:04 Test 2: INTEGER OPS W= 1.00 Reps= 500 T= 17.45ms
vovtasker Jan 27 17:49:04 Test 2: DOUBLE OPS W= 1.00 Reps= 25 T= 73.36ms
vovtasker Jan 27 17:49:04 Test 2: CHAR OPS W= 0.10 Reps= 10 T= 74.68ms
vovtasker Jan 27 17:49:04 ---- Weighted time: 98.27ms
vovtasker Jan 27 17:49:04 Test 3: INTEGER OPS W= 1.00 Reps= 500 T= 17.52ms
vovtasker Jan 27 17:49:04 Test 3: DOUBLE OPS W= 1.00 Reps= 25 T= 73.52ms
vovtasker Jan 27 17:49:04 Test 3: CHAR OPS W= 0.10 Reps= 10 T= 72.15ms
vovtasker Jan 27 17:49:04 ---- Weighted time: 98.26ms
vovtasker Jan 27 17:49:04 Best weighted time: 98.01ms
```

This method is very effective in discriminating among CPUs with the same architecture, such as all Sparcs. If you are running on a heterogeneous network, you will notice that the test routine executes

very well on PowerPCs and x86 processors compared to Sparc processors. To compensate this effect, we recommended setting the `tasker coefficient` on PowerPCs and x86 hosts to a value greater than 1.0 (e.g. 2.0).

On a multiprocessor machine, "raw power" refers to the power of each single CPU.

The *current power* of a machine takes into account both the raw power and the load (as computed by the UNIX utility `uptime`). On a multiprocessor, if the load is less than the number of CPUs minus one, the current power is the same as the raw power, because there is a processor that is practically idle.

The *effective power* takes into account the "self load" created by the jobs that have been initiated by the tasker itself.

Autokill of Jobs by vovtasker

If a job has the autokill field set to a positive value, the job will be killed by vovtasker/vovtasker if its duration exceeds the autokill value. The autokill field can be applied to any job of which the status is not Failed or Done.

The check for this condition is done by vovtasker every minute. This frequency is controlled by the "tasker update" value).

If a job is to be auto-killed, the vovtasker can use one of three methods:

- **Direct:** the tasker sends the job the signals TERM,HUP,INT,KILL. These can be overridden by `defaultStopSignalCascade` and `defaultStopSignalDelay` that can be set in `policy.tcl`, etc.

If `NC_STOP_SIGNALS` and/or `NC_STOP_SIG_DELAY` are set, then these will be used instead. The format of `NC_STOP_SIGNALS` can be a comma separated list. Each signal name such as "USR1", or the format `":SIGNAL:includerx:excluderx:skiptop"`

that is,

```
nc run -D autokill 10s -P
NC_STOP_SIGNALS=USR1::python:1,TERM,KILL -P
NC_STOP_SIGNAL_DELAY=1
```

send USR1 excluding any Python processes that are part of the job. Then send TERM and then KILL with a delay of 1s between signals.

`VOV_STOP_SIGNALS` and `VOV_STOP_SIGNALS_DELAY` work similarly to `NC_STOP_SIGNALS` and `NC_STOP_SIG_DELAY`

- **NC STOP:** the tasker calls `nc stop JOBID`; this only works for tasker that are running within Accelerator; this method honors the values of `NC_STOP_SIGNALS` and `NC_STOP_SIG_DELAY`.
- **VOVSTOP :** the tasker calls `vovstop -f JOBID` to kill the job.

The method used to autokill the jobs can be controlled on each tasker, meaning that all jobs on that tasker, if they need to be autokilled, will be killed with the same method.

Currently there is one way to change the autokill method on a tasker, and that is with this magic incantation:

```
% vovsh -x 'vtk_tasker_config TASKERNAME_OR_ID autokillmethod VALUE'
```

where VALUE can be one of the following keywords:

- direct
- ncstop
- vovstop

In reality, only the first letter of the keyword is used, i.e. 'd', 'n', 'v'. Anything else maps silently to 'direct'.

Examples:

```
% vovsh -x 'vtk_tasker_config lnx0123 autokillmethod ncstop'  
% vovsh -x 'vtk_tasker_config lnx0123 autokillmethod n'  
% vovsh -x 'vtk_tasker_config 00234567 autokillmethod direct'
```

Tasker RAM Saturation

The vovtasker implements the following features related to the exhaustion of memory on a tasker machine:

- RAM saturation occurs when the jobs executing on a tasker take up more memory than available on the machine so that the jobs may start swapping. For this computation use the "Virtual Memory" number and not the "Resident Set Size" number. When the tasker goes into saturation, the tasker orders all jobs by size and suspends enough jobs starting from the small ones to eliminate the saturation condition. The largest job is always left running, regardless of its size.
- If the free RAM on a tasker falls below a threshold, typically below 50MB, the tasker stops accepting jobs.
- The Ram Sentry mechanism, described in a separate section.

The behavior can currently be controlled only with the API `vtk_tasker_config` by specifying either the parameter `minRamFree` (Minimum Free RAM) or the parameter `effTotRam` (Effective Total RAM), both of which are expressed in MB (1MB = 1,048,576Bytes).

The default for `minRamFree` is 50 or 5% of the total RAM, whichever is less. Example:

```
% vovsh -x "vtk_tasker_config $taskerId minRamFree 20"
```

The default for `effTotRam` is the total RAM. Example:

```
% vovsh -x "vtk_tasker_config $taskerId effTotRam 16000"
```


Connect a Single Tasker

You can start a tasker at any time. Try the following commands and check the effect on the [taskers monitor](#).

```
usage: vovtasker [-A startupLogFile] [-a name] [-b capabilities] [-B]
                [-c coefficient] [-C cpus] [-d] [-D integer]
                [-e reserveExpr] [-E] [-f tclfile] [-F <file>]
                [-g taskergroup] [-G group] [-h host]
                [-H HEALTHCHECKFLAGS] [-i 0|1] [-I tclfile] [-j]
                [-k d|n|v] [-K] [-l rootOfDailyLogFile] [-L <loadSensor>]
                [-m <integer>] [-M max_load] [-n <integer>] [-N] [-o local
resources] [-p project] [-P <double>] [-q <hardbound>]
                [-Q <softbound>] [-r resources] [-R resources] [-s]
                [-S resources] [-t timeout] [-T capacity[/max_capacity]]
                [-U <CSV list of timeSpecs>] [-v number]
                [-V ncName@ncHost[:port]] [-w WX properties] [-W Tasker is
a member of a union.] [-x Experimental; do not use.]
                [-z <timeSpec>] [-Z <timeSpec>]

-A:          The name of startup log file
-a:          Name this tasker. The name may contain only letters, numbers,
dash(-) and underscore(_), or the expressions @HOST@ and @PID@
that get expanded on the fly
-b:          Comma-separated list of capabilities, case insensitive:
              symbolic: FULL NC LM
              normal   : PROCINFO NETINFO EXEC RT
              short    : P N X R

-B:          Show BPS tasker objects. Default to not show.
-c:          Tasker coefficient (positive, default 1.0)
-C:          Number of CPU's in this machine (automatic on win64). Use 0 to
specify default value.
-d:          Activate debugging
-D:          Min disk space in MB in /tmp and /usr/tmp (default 5)
-e:          Reserve tasker from the beginning: format of reserveExpr is
either the old 'GroupName/UserName/Duration' or the new list of
'keyword value' pairs where the keywords are USER GROUP
JOBCLASS JOBPROJ BUCKET ID DUR TRANSIENT. -e "JOBCLASS c1 DUR
1d TRANSIENT 1" will reserve the tasker for JOBCLASS c1 for 1
day. When the tasker disconnects, the reservation will be
removed as well
-E:          Use vovfire to execute jobs: valid for direct taskers only.
Disables caching of environments
-f:          Source the given Tcl file
-F:          ncTasker config file.
-g:          Set the taskergroup for this tasker
-G:          Specify Fairshare Group used by an indirect tasker. Use with -V.
-h:          Host (default is env. var. VOV_HOST_NAME)
-H:          Select health checks you want:
              P / p      Enable/Disable portmap check
              D / d      Enable/Disable disk space check
              W / w      Enable/Disable writability for /tmp and /usr/tmp
              U / u      Enable/Disable user script check
              ($VOVDIR/local/tasker/health_user_script.sh)
              Example: -H pDWu
-i:          Make the tasker transient (-i 1) or persistent (-i 0). If a
tasker is transient (the default), it is destroyed when the
client disconnects. Persistent taskers must be 'indirect'
taskers (see -I flag).
-I:          Indirect execution mode. The argument indicates the file that
```

describes the procedures to start and stop jobs indirectly. See the Reference Manual for more info. If the argument is just a dash '-' the option is ignored.

- j: Disable job statistics (useful on machines with lots of CPUS)
- k: Specify autokill mode (d=direct, n=ncstop, v=vovstop). Default 'd'
- K: Use Quick Connect
- l: Specify root of daily log file. The actual logs will be of the form FFFF_YYYY.MM.DD.log. Also closes stdin.
- L: Specify a SGE-like Load Sensor
- m: Maximum number of jobs allowed to run on tasker.
- M: Maximum allowed load on the tasker host. The default value for max_load is CAPACITY+0.5. Use 0 or less than 0 to specify default value. Use N or CAPACITY[+*/]N to make adjustments from the default. Examples: -M 12.0, -M CAPACITY+2, -M CAPACITY*2
- n: Run in nice mode with lower CPU priority
- N: Normal tasker. Same as -r @STD@
- o: Local resources (vovwxd internal)
- p: Project (default is env. var. VOV_PROJECT_NAME)
- P: Specify raw power of tasker, instead of computing it automatically.
- q: Dispatch only autokill job to hardbound tasker tasker.
- Q: Dispatch autokill and xdur job to softbound tasker tasker.
- r: Resources for this tasker. This can be either a list of resources or a Tcl expression that calls a procedure in the VovResources:: namespace. To simplify scripting, it is also possible to encode the resource string with base64 and pass the encoded string XXXX with -r base64:XXXX (i.e. no need to quote spaces in argument to -r option)
- R: Resources appended to the jobs by the agent. Use only with -V
- s: (OBSOLETE) Silent mode. Also closes stdin.
- S: Resource filter on what NC taskers to attach to the agent. Use only with -V
- t: Try multiple times to connect to server. Give up only after 'timeout' seconds.
- T: Specify capacity and optionally the max-capacity of the tasker. The capacity is the maximum number of jobs that can be run by tasker. The max_capacity is the maximum slots a tasker can be expanded to have when jobs are suspended. The default value for capacity is equal to the number of CORES present. The default value for max_capacity is 2*CAPACITY. Use -1 or 'auto' to specify the default. Use N, N/N, CORES[+*/]N, CORES[+*/]N/N, N/CORES[+*/]N, CORES[+*/]N/CORES[+*/]N to make adjustments from the default. Examples: -T 4, -T 4/8, -T AUTO/1000, -T CORES-2, -T CORES*0.8, -T CORES+0/20, -T CORES+2/CORES*2
- U: Update intervals for resources, tasker statistics, and job statistics. The resources update interval is also known as the tasker heartbeat. One (resources), two (resources,taskerStats), or three (resources,taskerStats,jobStats) values may be specified. Defaults: 60s,120s,30s
- v: Set verbose level (0-4): default 1.
- V: NetworkComputer indirect tasker
- w: Reserved for system use
- W: Reserved for system use
- x: Experimental; do not use.
- z: After being idle for given time, tasker exits.
- Z: After specified lifetime, tasker does not accept new jobs and exits after completing active jobs.

Tasker Status

The following table applies to "direct" taskers and to "BPS agents".

Status	Color	Description
Idle	green	The tasker is ready to run jobs.
Working	dark yellow	The tasker is running some jobs and it has some additional capacity.
Full	light yellow	The tasker is running as many jobs as it is capable.
Overloaded	red	The load on the host where the tasker is running is higher than the maximum load allowed by the tasker.
Busy	light gray	The tasker is doing some bookkeeping operation.
Suspended	purple	The tasker has been suspended and it is not accepting any new jobs.
Paused	pale purple	The tasker has been paused by a signal from an external process. It can be awakened by sending a SIGCONT signal.
Sick	black	The tasker has not sent the heartbeat for more than twice its update interval (update interval is 1 minute by default).
Requested (previously called Down)	dark gray	The tasker is persistent (not transient) and the associated client is currently not connected.
Done	blue	The tasker is about to exit and no longer accepting jobs.
NoLic	gray	The tasker is waiting for a license.

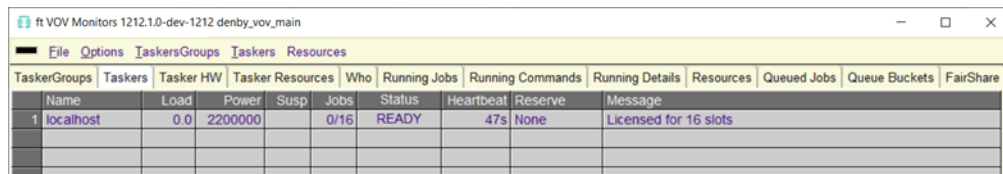
Status of BPS Taskers

A BPS tasker is a tasker that represent a computing host that belongs to another Batch Processing System (e.g. SGE or Accelerator).

Status	Color	Description
Ok	green	Not in an error condition.
Warning	red	Used for BPS taskers. The BPS system reports some problems with this tasker.
Unknown	gray	The BPS system has not reported enough information about this host.

Monitor Taskers

To monitor the activity of the taskers, use either `vsm` from the CLI, or click **Console > Vov Monitor** from the GUI.



The screenshot shows a web application window titled "ft VOV Monitors 1212.1.0-dev-1212 denby_vov_main". It has a menu bar with "File", "Options", "TaskersGroups", "Taskers", and "Resources". Below the menu is a tabbed interface with tabs for "TaskersGroups", "Taskers", "Tasker HW", "Tasker Resources", "Who", "Running Jobs", "Running Commands", "Running Details", "Resources", "Queued Jobs", "Queue Buckets", and "FairShare". The "Taskers" tab is active, displaying a table with columns: Name, Load, Power, Susp, Jobs, Status, Heartbeat, Reserve, and Message. The first row shows a tasker named "localhost" with a load of 0.0, power of 2200000, 0/16 jobs, a status of READY, a heartbeat of 47s, and a message "Licensed for 16 slots".

Name	Load	Power	Susp	Jobs	Status	Heartbeat	Reserve	Message
1 localhost	0.0	2200000		0/16	READY	47s	None	Licensed for 16 slots

Figure 5:

From the command line, you can use:

```
% vovtaskermgr show
1 04204992  bellevue 0.00    90909 0/1  READY  Unlim.
2 04205717    mars 0.00    39239 0/1  READY  Unlim.
3 04205719   saturn 0.05    10633 0/1  READY  Unlim.
4 04205720    moon 0.02    15003 0/1  READY  Unlim.
5 04205721   cayman 0.00    83333 0/1  READY  Unlim.
6 04205722   comet 0.07    58595 0/1  READY  Unlim.
7 04205725  cheetah 0.05    224089 0/1  READY  Unlim.
8 04205726   jupiter 0.03    13404 0/2  READY  Unlim.
```

From the browser interface, go to the Tasker page. For a large number of taskers, such as hundreds or more, you may find the Tasker page more compact and convenient.

Tasker Resources

The resources offered by a tasker are represented by a space-separated list of tokens. Resources can be **explicit** or **symbolic**.

The resources described in this section have been obsoleted. The new information about Hardware Resources is provided in Hardware Resources. The resources described in this section are still supported.

The resource list is the concatenation of two sublists:

- The resources specified with option `-r`, which can be either a static list (explained below) or dynamically computed list as explained in [Time-Variant Taskers](#).
- The resources specified in the `taskerClass.table` file.

The option `-r` in `vovtasker` defines the resources offered by the tasker.

Examples of explicit resources:

```
% vovtasker -r "unix diskio RAM/512"  
% vovtasker -r "RAM/3000 REGRMACHINE"
```

Symbolic Resources for Taskers

The following table describes all symbolic resources defined for taskers. By default, a tasker provides the resources corresponding to the symbolic resource `@STD@`.

Symbolic Resource	Description
@ARCHITECTURE@	It has value 'win64' on Windows, and the same as @MACHINE@ on UNIX.
@CPUS@	A consumable resources indicating the number of CPUS in the tasker (example for a 4-CPU machine: "CPUS/4").
@DISPLAY@	The name of the X display accessible by the tasker and derived from the value of the environment variable DISPLAY.
@HOST@	The name of the host on which the tasker runs.
@MACHINE@	The name that identifies the machine type . On Windows, this have value 'x86'.
@MODEL@	Linux: an abbreviated version of the "model name" line in the output of <code>/proc/cpuinfo</code> , otherwise the value is "Model:unknown". Not available on the other platforms.

Symbolic Resource	Description
@OS.VERSION@	The name of the operating system, including the version number (for example, Linux.1.0).
@OS@	The operating system name (for example, Linux).
@OSCLASS@	Either "unix" or "windows".
@RAMFREE@	<p>The amount of available physical memory (RAM), in MB. On a Linux system, this is computed by opening <code>/proc/meminfo</code> and by adding up the values for MemFree, Buffers, and Cached.</p> <pre>% cat /proc/meminfo MemTotal: 6100392 kB MemFree: 1848576 kB Buffers: 188596 kB Cached: 2686368 kB SwapCached: 16 kB Active: 1847404 kB ...</pre>
@RAMTOTAL@	The total amount of physical memory (RAM), in MB.
@RAM@	This is a consumable resource representing the residual amount of RAM after taking into account the jobs running on the tasker.
@RELEASE@	This is mostly used on Linux to represent the name of the Linux distribution. On most systems, this is computed by running <code>lsb_release -isr</code> .
@SMARTSUSPEND@	Either the string "smartsuspend" if SmartSuspend (by Jaryba) is available on the machine or the null string "".
@STD@	The default for each tasker, which corresponds to the set @CPUS@ @DISPLAY@ @USER@ @HOST@ @RAM@ @MACHINE@ @VOVARCH@ @OSCLASS@ @OS.VERSION@ @VIEW@ @RELEASE@.
@SWAPFREE@	The amount of available swap space, in MB.
@SWAPTOTAL@	The total amount of swap space, in MB.

Symbolic Resource	Description
@SWAP@	This is a consumable resource representing the residual amount of SWAP after taking into account the jobs running on the tasker.
@USER@	The name of the user that owns the tasker. This is the value of either the variable LOGNAME or of the variable USER.
@VIEW@	The view name as defined by ClearCase.
@VOVARCH@	The value of the environment variable VOVARCH, normally set with <code>vovarch</code> .

The taskerClass.table File

This file allows the classification of taskers without requiring the taskers to be restarted. The file `taskerClass.table` resides in the server configuration directory (that is, the directory `$VOV_PROJECT_NAME.swd`). Each tasker checks the file for changes about once a minute.

The file consists of lines with the following format:

```
<taskername>: <resource list>
(default): <resource list>
```

All lines that do not conform to this format are ignored. The `taskername` begins at the beginning of the line. If the tasker name is the string `(default)`, the line indicates the default value of the resource list. Note that `(default)` is not a legal tasker name because it contains the characters `()` which are not allowed in tasker names.

```
# Example of taskerClass.table file
(default): classC

ferrari: classA classB
buick: classA
dodge: classB
```

Time-Variant Taskers

Time-variant taskers resources can be configured using Tcl. Any procedure can be defined in the namespace `VovResources` and then used to compute the resource list.

If the first characters in the argument for the option `-r` is the sequence `VovResources::`, then the taskers resources are computed by executing the argument as a Tcl command.

For a list of predefined procedures, refer to *Predefined VovResources:: Procedures*.

A special case of a resource is the local disk space on a host, which varies according to the files stored there. There may be jobs that require a minimum amount of space to run successfully. The vovtasker implements `vtk_fs_stat` (see `vtk_fs`) to handle such requirements. An example is provided further below.

An alternative is to use load sensor. However, in the case of free disk space, `vtk_fs_stat` is recommended because it is more efficient.

For example, taskers can be set up to offer one set of resources during the day and another set of resources during the night. The following example is a script that computes the resource "nothing" between 5am and 8pm, and the standard resources during the night. In addition, `vtk_tasker_set_timeleft` is used to control the maximum expected duration of the jobs sent to the tasker.

```
namespace eval VovResources {
    proc Night { args } {
        # Return the standard resources during the night and suspend the tasker
        # during the day. During the night, the tasker progressively reduces the
        # maximum length of the jobs it accept.
        set NIGHT_RESOURCES "@STD@ $args"
        set EVENING_START 19
        set MORNING_END 6
        set HH [clock format [clock seconds] -format "%H"]
        regsub {^0} $HH "" HH; # Strip leading 0.
        if { $HH >= $MORNING_END && $HH < $EVENING_START } {
            # -- During the day, suspend the tasker.
            vtk_tasker_set_timeleft 0
        } else {
            # -- During the night, compute the time left.
            set hoursLeft [expr $MORNING_END - $HH]
            set hoursLeft [expr $hoursLeft >= 0 ? $hoursLeft : $hoursLeft + 24]
            set timeleft [expr $hoursLeft * 3600]
            vtk_tasker_set_timeleft $timeleft
        }
        return $NIGHT_RESOURCES
    }
}
```

Example:

```
% vovtasker -r "VovResources::Night hspice"
```

These procedures are evaluated:

- Every minute, or at the interval that was selected with the option -U
- After the completion of each job.

The standard procedures are defined in the script `$VOVDIR/etc/tasker_scripts/taskerRes.tcl`. It is recommended to review these procedures before implementing your own procedures in `$VOVDIR/local/taskerRes.tcl`.

Free Disk Space

The following script monitors free disk space using `vtk_fs_stat`. This example script adds the resources `WORK` and `SCRATCH`; the values of these resources will be the amount of disk space in MB on the corresponding filesystem.

```
namespace eval VovResources {
    proc FsCheck { args } {
        # Usage:
        #   VovResources::FsCheck -fs WORK /work -fs SCRATCH /scratch -r @STD@ -r xx
        #
        set resources {}
        while { $args != {} } {
            set arg [shift args]
            switch -- $arg {
                "-fs" {
                    set name [shift args]
                    set dir [shift args]

                    set space [vtk_fs_stat $dir]
                    lappend resources "$name#$space"
                }
                "-r" {
                    lappend resources [shift args]
                }
            }
        }
        return [join $resources]
    }
}
```

Example:

```
% vovtasker -r "VovResources::FsCheck WORK /work"
```

Predefined VovResources:: Procedures

This section describes the predefined procedures that can be used to characterize the taskers in your farm in a time-variant fashion. To use these procedures, you have to use them as resources for a tasker.

An example of using `VovResources::` in `taskers.tcl`:

```
vtk_tasker_define linux102 -resources "VovResources::Offhours -evening 20 -morning 7"
```

The standard procedures are defined in `$VOVDIR/etc/tasker_scripts/taskerRes.tcl`. You are encouraged to study these procedures and to implement your own procedures in `$VOVDIR/local/taskerRes.tcl`.

Procedure	Arguments	Description
Custom	args	Just returns \$args

Procedure	Arguments	Description
Standard	args	Just returns @STD@ \$args
FsCheck	args	<p>This procedure checks the disk space on some filesystems. It is based on the procedure <code>vtk_fs_stat</code>. It supports two options:</p> <ul style="list-style-type: none"> • <code>-r Resource</code> to specify a resource of the tasker; it may be repeated • <code>-fs NAME Directory</code> to specify the name of a filesystem resource and a directory in that filesystem; it may be repeated <p>Example:</p> <pre>VovResources::FsCheck -fs SCRATCH / export/scratch -r @STD@</pre>
Workstation	minIdle maxTime args	<p>This is the old and now obsolete syntax for this policy. Please see below for the updated syntax. This procedure checks the time from the last interaction with the mouse or the keyboard. If it is less than <code>minIdle</code>, we assume that the workstation is not idle and we suspend the tasker. If the workstation is idle, we allow jobs with expected duration less than <code>maxTime</code>. Arguments <code>minIdle</code> and <code>maxTime</code> are Time Specifications.</p>
Workstation	[options]	<p>Options are:</p> <ul style="list-style-type: none"> • <code>-resources "list"</code> -- Specify the list of resources. • <code>-minidle time</code> -- Specify minimum time the tasker has to be idle (no keyboard or mouse activity) for the tasker to become active. • <code>-maxtime time</code> -- Specify max expected duration of jobs sent to tasker. • <code>-suspend</code> -- By default, when the tasker becomes inactive, the current jobs are completed and no new jobs are accepted. With this option the current jobs are suspended.
Night	args	Return the standard resources during the night and suspend the tasker during the day. During the night, the tasker progressively reduces the maximum length of the jobs it accepts.

Procedure	Arguments	Description
Offhours	args	<p>Return the standard resources during off hours (night and weekends) and suspend the tasker during the rest of the time. During off-hours, the tasker progressively reduces the maximum length of the jobs it accepts. Supports the following options:</p> <p><i>-owner user</i> To specify a user that has overriding control on this tasker (the user would use <code>vovtaskerowner</code>). You also need to specify the <code>-host</code> option.</p> <p><i>-host host</i> Required if you use -owner. Specify the name of the host to suspend the jobs when the tasker is suspended.</p> <p><i>-susp</i> To suspend the jobs when the tasker is suspended.</p> <p><i>-evening hour</i> The hour (0-23) at which the tasker starts accepting jobs. The default is 19.</p> <p><i>-morning hour</i> The hour (0-23) at which the tasker stops accepting jobs. The default is 6.</p>

Tasker Policies

Tasker Reservations

A vovtasker may be reserved for one or more:

- Users
- FairShare groups
- OS groups
- Jobclasses
- Job projects
- Jobs
- Buckets

The reservation is always for a specific period of time, starting at any time and ending some time in the future. If the end time is in the past, the reservation is ignored and removed. The reservation can be very long, for example thousands of days.

A tasker can have multiple reservations at any time. Jobs that do not match any reservation are not sent to the tasker if the tasker is reserved. For example:

- If the tasker is reserved for a user, only jobs submitted by that user may be dispatched to that tasker.
- If the tasker is reserved for a group, only jobs from that group may be dispatched to that tasker.
- If the tasker is reserved for a user and a group, only jobs submitted by that user who belongs to that group may be dispatched to that tasker.

Negated Reservations

Taskers may also be reserved for a negated set, in order to prevent access to the tasker by jobs that match members of the set. To negate a tasker reservation, simply place a not symbol "!" at the beginning of the reservation expression. For example, the command `vovtaskermgr reserve -user '!john,mary' -duration 1h tasker1` will allow the tasker named `tasker1` to run jobs from any user except `john` and `mary` for 1 hour. Any valid reservation expression can be negated.



Note: If you are negating a list, use the not symbol ONLY at the beginning of the list; you should NOT place a "!" before every item.

Persistent Reservations

Reservations are persistent. If you stop a tasker and restart another one with the same name, the new tasker will inherit the persistent reservation of the old tasker. Expired reservations are removed frequently.

If a tasker is renamed, the corresponding reservations get updated too. So, the reservations do not get lost.

Reservation rules

Only reservations with end time later than the current time are valid.

There can be multiple reservations on a vovtasker. But there is only one active reservation.

If there is already an active reservation, a new reservation overlapping time with the active reservation can be created but ignored. When the active reservation expires (end time passes current time), a new active reservation with earliest start time is picked.

Reservation only applies to normal (direct) taskers or BPS agents. Making reservations on indirect taskers is not allowed.

Duplicate reservations

A new reservation can be created if the reservation is different from existing reservations. Two reservations are the same if the following attributes of reservations are the same. All of these attributes can be specified when creating a reservation using `vtk_reservation_create`. If a reservation is created through `vovtaskermgr`, `vtk_tasker_reserve`, and `taskers.tcl`, `start time` and `end time` are not used as identifiers. The existing reservation with the same other attributes are updated with the new start time and end time. If there is no existing reservation, a new reservation is created.

<code>type</code>	Always <code>tasker</code> if the reservation is for tasker.
<code>what</code>	List of tasker names that this reservation is reserving
<code>start time</code>	Reservation start time
<code>end time</code>	Reservation end time
<code>user</code>	Reservation is for these users
<code>group</code>	Reservation is for these groups
<code>osgroup</code>	Reservation is for these osgroups
<code>jobclass</code>	Reservation is for these jobclasses
<code>jobproj</code>	Reservation is for these jobprojects
<code>bucketid</code>	Reservation is for these bucket IDs
<code>id</code>	Reservation is for these job IDs

Reserve a vovtasker

There are several ways to create a new reservation.

`vovtaskermgr`, `taskers.tcl`, and `vtk_tasker_reserve` creates a new reservation, but if there is an overlapping reservation with the same parameters, the existing one is updated with new `start_time` and `end_time`. These are the same with running `vtk_reservation_create` with the `-update` option.

On the Command Line: vovtaskermgr reserve

The syntax is:

```
% vovtaskermgr reserve [options] <taskers_list>
```

where the options could be:

-user	Comma separated list of users
-group	Comma separated list of groups
-jobproj	Comma separated list of job projects
-jobclass	Comma separated list of job classes
-osgroup	Comma separated list of OS groups
bucketid	Comma separated list of bucket IDs
-id	Comma separated list of job IDs
-start	Start time
-end	End time
-cancel	
<tasker_list>	List of tasker names to reserve.

For example, the following command reserves taskers `jupiter` and `alpaca` for user `john` for 3 hours starting from now.

```
% vovtaskermgr reserve -user john -duration 3h jupiter alpaca
```

With `-cancel` option, all reservations on the specified tasker(s) will be removed.

```
% vovtaskermgr reserve -cancel jupiter alpaca
```

The following command shows all reservations for taskers.

```
% vovtaskermgr reserveshow
```

In the taskers.tcl File

This is useful to reserve a tasker from the instant it is created. Use option `-reserve` of `vtk_tasker_define`.

The reservation expression argument to the `-reserve` option takes space-separated list of key value pairs, where the key is one of `USER`, `GROUP`, `JOBCLASS`, `JOBPROJ`, `BUCKET`, `DUR`, `QUANTITY`. If the key is `DUR`, the value is a time spec. If not specified, the default duration is 1 year. For the other keys, the value is a comma-separated list.

Examples:

```
vtk_tasker_define jupitar -reserve "USER john,mary JOBCLASS spectre"  
vtk_tasker_define alpaca -reserve "JOBPROJ chipa,chipb DUR 3w"
```

The old form GROUP/USER/DURATION is accepted. The GROUP and USER parts are optional, but the separators ('/', the forward-slash character) must be present. The duration is expressed as a VOV timespec, e.g. 2d for two days. If only digits are present, the value is interpreted as seconds.

-reserve is passed to vovtasker executable as -e option. If advanced users want to run a tasker with -e option for initial reservation, the syntax is the same as vtk_tasker_define -reserve option.

Reserving a Tasker via the Browser

Open the tasker page to reserve a particular tasker. It is at the URL /tasker/taskerId. Fill out a simple form, indicating which user, group, OS group, job class, and/or job project for this tasker is to be reserved, as well as start time and duration. After you select the duration, the form will be submitted.

Click **Forget** to cancel the tasker reservation, if you are ADMIN.

vtk_tasker_reserve Tcl Interface

With Tcl, you can use tasker reservations. The syntax is:

```
vtk_tasker_reserve taskerId [-user <user1,user2,...>]  
                             [-group <group1,group2,...>]  
                             [-osgroup <osgroup1,osgroup2,...>]  
                             [-jobclass <jobclass1,jobclass2,...>]  
                             [-jobproj <jobproj1,jobproj2,...>]  
                             [-bucketid <bucketid1,bucketid2,...>]  
                             [-id <jobid1,jobid2,...>]  
                             [-start <starttime>]  
                             [-end <endtime>]  
                             [-duration <reserved_duration>]
```

For example, the following line will clear (cancel) all reservations on tasker 00001230, if there is one. Otherwise, this doesn't have any effect.

```
vtk_tasker_reserve 00001230
```

The following line reserves tasker 00001230 for user john for 3 hours starting from now.

```
vtk_tasker_reserve 00001230 -user john -duration 3h
```

This reserves tasker 00001230 for user john in group alpha for 2 weeks starting from one hour from now.

```
vtk_tasker_reserve 00001230 -user john -group alpha -start [clock scan "1 hour"] -  
duration 2w
```

vtk_reservation_create Tcl Interface

You can use vtk_reservation_create in a Tcl interface. This is a new interface introduced in 2017 version to support multiple reservations per tasker.

The syntax is:

```
vtk_reservation_create <type> <what> <quantity> <start_time> <end_time> [options]
```

where:

<what>	Comma separated list of tasker names
<quantity>	Not used for tasker reservations
<start_time>	Reservation start time
<end_time>	Reservation end time

And the options could be:

-user	Comma separated list of users
-group	Comma separated list of groups
-osgroup	Comma separated list of OS groups
-jobclass	Comma separated list of jobclasses
bucketid	Comma separated list of bucket IDs
-id	Comma separated list of job IDs
-update	

For example, the following Tcl script creates a reservation for host1 and host2. It reserves 4 slots of each host. It returns ID for the reservation. The ID can be used to update and delete the reservation.

```
set now [clock seconds]
vtk_reservation_create tasker host1,host2 1 $now [expr $now+3600] -user brian
```

If all attributes are the same as one of existing reservations, `vtk_reservation_create` will return `nochange`.

With `-update` option, it looks for a reservation which has same attributes except `start_time` and `end_time` but the reservation period is overlapping. If there is one, the existing reservation is updated with `start_time` and `end_time`.

```
set now [clock seconds]
set end [expr $now+3600]
set end2 [expr $now+7200]
vtk_reservation_create tasker localhost 1 $now $end -user john #creates a new
reservation
vtk_reservation_create tasker localhost 1 $now $end -user john #returns "nochange"
vtk_reservation_create tasker localhost 1 $now $end -group g1 #creates a new
reservation
vtk_reservation_create tasker localhost 1 $now $end2 -user john -update #updates the
first
```


Manage Reservations

To show all existing reservations, use:

```
% vovshow -reservations
```

To forget all reservations, use:

```
% vovforget -allreservations  
% vovforget <reservationId>
```

Reservations data is accessible with `vovselect` as well.

```
% vovselect id,reserveuser,reservestart,reserveend from reservations
```

vtk_reservation_get Tcl Interface

`vtk_reservation_get <reservationId> <variable>` provides the details about a reservation.

```
vtk_reservation_get 21673 info  
parray info
```

Prints out the following information:

```
info(id)           = 21673  
info(quantity)     = 1  
info(reservebucketid) =  
info(reservecreated) = 1513026518  
info(reservedby)    = jin  
info(reserveend)    = 1513199318  
info(reservegroup)  =  
info(reserveid)     =  
info(reservejobclass) =  
info(reservejobproj) =  
info(reserveosgroup) =  
info(reservestart)  = 1513026518  
info(reserveuser)   = jin  
info(type)          = tasker  
info(what)          = local2
```

`vtk_reservation_update <reservationId> <fieldname> <new_value>` updates a field of reservation with a new value. Available field names can be found by `vovselect fieldname from reservations` on the command line.

```
vtk_reservation_update 21673 RESERVEUSER robert
```

`vtk_reservation_delete <reservationId>` removes the reservation.

```
vtk_reservation_delete 21673
```

Number of Reservations and System Performance

Many reservations on each tasker may slow down the system. Upon choosing the right tasker to run a job, the algorithm considers all reservations. It is recommended to use the limited number of reservations per tasker. By default, the maximum number of reservations per tasker is set as 10 and this is configurable through a server parameter `tasker.max.reserve` in `policy.tcl`.

Tasker Owner

The program `vovtaskerowner` gives the owner of a machine the control on the tasker running on such machine.

By invoking `% nc cmd vovtaskerowner &` you get the dialog shown below.

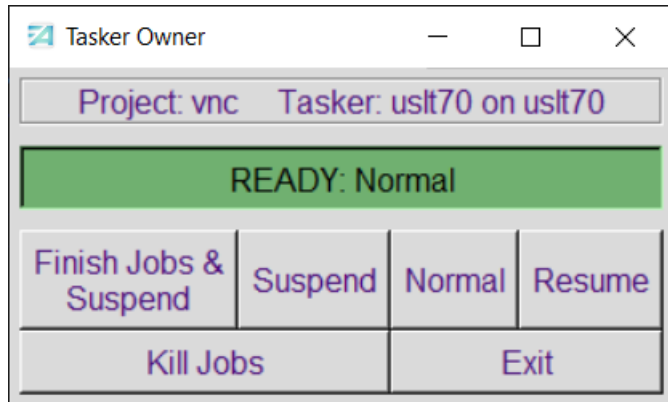


Figure 6:

Suspend suspends the tasker immediately for one hour. Each time Suspend is clicked, the tasker is suspended for an additional hour.

Resume resumes the activity of a suspended tasker for an hour. Each time Resume is clicked, the tasker remains active for an additional hour.

Normal returns the tasker to the normal behavior.

This dialog maintains its status in a file called `~/.vnc_<TASKERNAME>`. This is also the file that is being read by the `vovtasker` binary.

```
vovtaskerowner: Usage Message

DESCRIPTION:
    vovtaskerowner [OPTIONS]

OPTIONS:
    -v                -- Increase verbosity
    -h                -- This help
    -taskername NAME  -- Specify the name of the tasker we want to control.

EXAMPLES:
    % vovtaskerowner -h
    % vovtaskerowner &
    % vovtaskerowner -taskername mac05 &
```

Administrator Responsibilities

The mechanism shown in this page works only if the tasker is using the [time-variant resources](#) "VovResources::Offhours"

```
# Fragment of taskers.tcl file.  
# This line allows user johnny to be considered the "Owner" of# the tasker on machine  
linux123  
vtk_tasker_define linux123 -resources "VovResources::Offhours -owner johnny"
```

Taskers: Mixed Windows NT and UNIX Environment

It is common to run VOV in a mixed network of UNIX and Windows NT workstations. Following is a set of guidelines to facilitate the deployment in such a network.

To avoid case-based ambiguities with file names, use only lowercase names in your design.

Define all appropriate [equivalences](#) in the `equiv.tcl` file. For example, if a project directory is mounted as `/projects/local/top` in UNIX and as `p:/local/top` on Windows NT, you can define the following equivalences:

```
vtk_equivalence TOP /projects/local/top  
vtk_equivalence TOP p:/local/top
```

In the `taskers.tcl` file, be explicit about specifying the directory of the VOV installation as seen from the Windows NT machines as well as the working directory of the server as seen from the Windows NT machines. Example:

```
vtk_tasker_set_default -vovdir v:/vov/winnt -serverdir p:/local/  
top/vovadmin  
vtk_tasker_define nthost1vtk_tasker_define nthost2
```

The tool `vovtaskermgr` normally uses a remote shell (either `rsh`, `remsh` or `ssh`) to start taskers on remote UNIX machines. Windows NT does not offer an effective equivalent to the UNIX remote shell, so you can either start taskers manually on each Windows NT workstation, or you can use [vovtsd](#).

Start a Remote UNIX TASKER Details

On UNIX, the script `vovtaskerstartup` is used to start a tasker on a remote machine.

This script ensures that the tasker runs in a valid environment by sourcing the following scripts:

- `$VOVDIR/local/scripts/vovtaskerstartup.aux`, if available, to perform site specific initialization:

```
# Example of $VOVDIR/local/scripts/vovtaskerstartup.aux  
echo "This is vovtaskerstartup.aux on `uname -n`"  
switch ( $VOVARCH )  
case "linux*":  
    unlimit openfiles  
    breaksw  
default:
```

```
endsw
```

- `$VOVDIR/etc/std.vov.aliases`, to define all standard aliases (for example, `ves`)
- `setup.tcl` in the server configuration directory to initialize the project environment.

If the `-view` option is used, the script also starts the tasker in the given ClearCase view.

The `-descriptors` option triggers a check to ensure the host has at least as many descriptors as `vovserver`, which ensures it can operate at full capacity in the event of failover. It also checks that the `server_election` directory is empty.



Note: For this function to work, the `-failover` option must be used with `vtk_tasker_define`. For more information, refer to [vtk_tasker_define](#).

Stop Taskers

When you no longer need a tasker that you started manually and that is running in the foreground, stop it with `Ctrl-C`.

Background taskers, and taskers started with `vovtaskermgr`, can be shut down forcibly with `vovkill pid` or more cleanly with

```
% vovtaskermgr stop  
% vovtaskermgr stop <tasker_name>
```

Clean Up

Occasionally, the taskers leave some files in `/usr/tmp`. You can use `vovcleanup -host` to browse for files left over by the taskers.

```
% vovcleanup -host
```

Taskers Running as Root: Security Implications

A normal `vovtasker`, as described in [Taskers](#), has the privileges and limitations of the account in which it runs. When a normal `vovtasker` executes a command, the effect is the same as when the `vovtasker`'s owner runs the command.

For Accelerator, jobs may be submitted by many users. To obtain the correct permissions with respect to files needed by the job, the `vovtasker` must be able to switch to the account of the submitter. On UNIX and Linux machines, this is done by having `vovtasker` run with root privilege. A simple method is creating a copy of `vovtasker` called `vovtaskerroot`, which is owned by root and has the `setuid` flag set. `vovtaskerroot` can be set up by running the script `SETTASKERUID.cshS`.

To verify that `vovtaskerroot` is properly installed, go to the `$VOVDIR/bin` directory and check the access flags of `vovtaskerroot`. Verify the "s" character is in the fourth column (instead of the "x" character) as shown below:

```
% cd $VOVDIR/bin
% ls -l vovtaskerroot
-rwsr-sr-x  1 root    other      875092 Jan  5 11:57 vovtaskerroot
```

For security reasons:

- The `USER` field for each job cannot be edited by any user, including `ADMIN`.
- The `vovtasker` itself never executes any process as root. Instead, each job is executed as the user that first executed the job. If the user account does not exist on the tasker host, the job cannot be executed.


Other Methods of Starting vovtasker with Root Capability

Some sites may have security policies that prohibit `setuid-root` binaries, or prohibit `setuid` binaries from being mounted over NFS. Following are alternate methods to start `vovtasker` with root capability:

- Start `vovtasker` from an `init.d` script.
- Use a `setuid vovtaskerroot` on a local disk.

Using a setuid vovtaskerroot Binary on a Local Disk

If the NFS filesystem (including the Altair Accelerator) is exported with the `nosuid` option, the `vovtaskermgr` command can still be used via a local `setuid-root` binary on each farm host.

 **Note:** This method will add the cost of having to update the hosts individually when changing versions.

For example, the binary can be put at `/opt/rtda/some-version/linux/bin/vovtaskerroot`. It is helpful but not necessary if this path is the same on all farm hosts.

In this case, the `setuid-root` binaries must be created manually (the regular script is not useful). Following is an example of creating the `setuid-root` binary:

```
% ssh some-farm-host
% /bin/su -
# cd /opt; mkdir -p rtda/CURRENT/linux/bin
# cd /opt/rtda/CURRENT/linux/bin
# cp /network-path-to-rtda/CURRENT/linux/bin/vovtasker ./vovtaskerroot
# chown root: ./vovtaskerroot
# chmod u+s ./vovtaskerroot
```

After creating the local `vovtaskerroot`, set up the taskers configuration file of Accelerator to use the local `vovtaskerroot`. The file is located in `$VOVDIR/../../vnc/vnc.swd/taskers.tcl`.

If the path to the local `vovtaskerroot` binary is the same on all farm hosts, change the defaults at the beginning of the file as shown below:

```
if { [file executable /opt/rtda/CURRENT/linux/bin/vovtaskerroot] } {
    # Use vovtaskerroot from the local disk
    vtk_tasker_set_default -executable /opt/rtda/CURRENT/linux/bin/vovtaskerroot
} else if { [file executable $env{VOVDIR}/bin/vovtaskerroot] } {
```

```
    vtk_tasker_set_default -executable vovtaskerroot
} else {
    vtk_tasker_set_default -executable vovtasker
}
```

If the path to the vovtaskerroot varies from host to host, add the -executable option to the definition for each host (instead of changing the default).

Server Impersonation on Windows

In Windows, there is no superuser identity such as root that can switch to other accounts without providing any credentials. For Accelerator on Windows, a different method is used to run jobs as the submitter.

In Windows, the vovtasker calls a Windows API to create a process with a username and password. The username is mapped from the UNIX/Linux username, and the password is stored in encrypted form in the vovserver and passed to the vovtasker. The password is destroyed immediately after use. It is possible for a UNIX/Linux user to run jobs as another Windows user, by logging onto the machine locally with the account name and password.

After the Windows process is running as the correct user, it may need to mount any filesystems needed, as each user has their own set of drive letters in recent Windows versions. For details, refer to *Vov Windows Impersonation* in the Altair Accelerator Administration Guide.

vtk_job_control

From Tcl, you can control both the taskers and the jobs running on the remote taskers with the procedure `vtk_job_control`.

This is the main procedure to send controlling signals and modifications to jobs running on remote taskers. The argument `taskersId` 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 tasker are affected.
- If `jobId` is the string "TASKER", then the tasker is stopped gracefully.
- If `jobId` is the string "TASKER/FORCE", then the tasker is stopped with force. Taskers can only be stopped.

Table 1: Options

Option	Action EXT	Action MODIFY
opt1	signalName	fieldName
opt2	procNameIncludeRx	newValue
opt3	procNameExcludeRx	unused



Note:

- STOP and KILL are equivalent
- SUSPEND may use colon delimiters to specify which signals to use and include/exclude lists, formatted as:
SUSPEND:[comma-delimited signal list]:[comma-delimited include list]:[comma-delimited exclude list] . See Examples.
- STOP may use colon delimiters to specify which signals to use, include/exclude lists and delay in seconds between sending signals, formatted as:
STOP:[comma-delimited signal list]:[comma-delimited include list]:[comma-delimited exclude list]:[delay between signals in seconds] . See Examples.
- 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 of vtk_job_control

```
# Stop all taskers
vtk_job_control ALL STOP TASKER

# Stop tasker 000123456 with force
vtk_job_control 000123456 STOP TASKER/FORCE

# Stop all jobs on tasker 000123456
vtk_job_control 000123456 STOP 0

# Stop job 223344 on tasker 000123456
vtk_job_control 000123456 STOP 223344

# Stop job 223344 running on any tasker
vtk_job_control ALL STOP 223344

Suspend all sleep jobs using SIGINT followed by SIGHUP on tasker 12345:
vtk_job_control 12345 SUSPEND:INT,HUP: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::sleep:2
```

Interfaces to Accelerator, SGE, LSF, and Others

FlowTracer can be used also as a front end to an existing batch processing system (BPS), for example, Accelerator, LSF, SGE, SGEEE, etc.

Connect to a Third Party Batch Processing System

FlowTracer provides default implementations of interfaces for some of the Batch Processing Systems. FlowTracer also provides Tcl APIs for easy, flexible and extensible implementation of interfaces to the Batch Processing System of your interest. This enables FlowTracer to fit into any existing Enterprise Grid.

Architecture Diagram

The following diagram shows how FlowTracer projects work with several Batch Processing Systems. In this example, they are Accelerator and SGE.

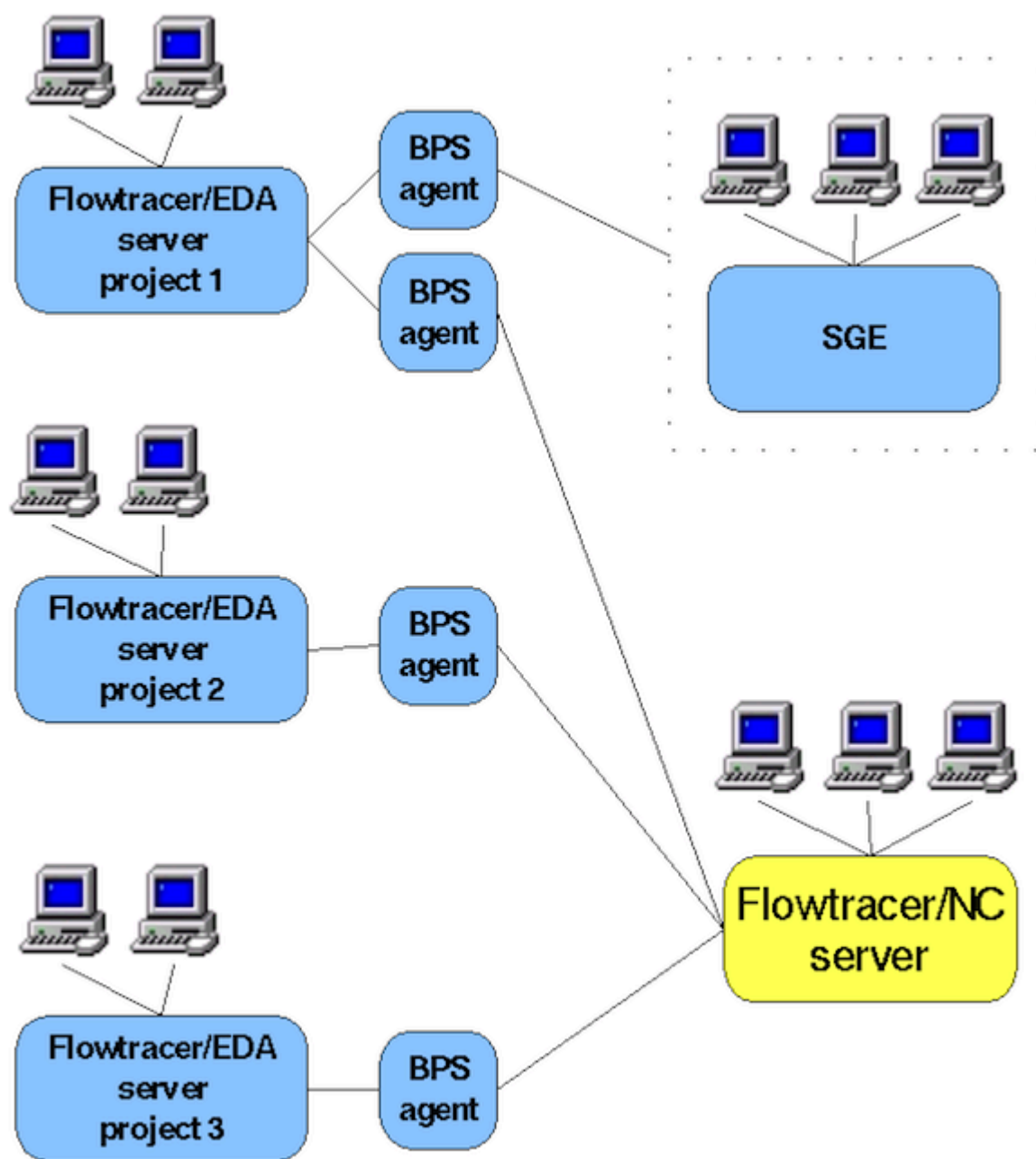


Figure 7:

Batch Processing System Agent

You can create a Batch Processing System agent as seen in the diagram as the **BPS agent**. It is the interface that submits jobs to the Batch Processing System that it gets from a FlowTracer server, instead of executing them directly.

Interface FlowTracer with Accelerator

There are two ways to interface FlowTracer to Accelerator.

They are:

- With a **binary interface** capable of supporting multiple users. The jobs are injected directly into the memory of the Accelerator server, thus preserving safely the credentials (user, osgroup) of each job. This method bypasses the policy layers implemented as part of the `nc run` command, which may be a disadvantage in some cases.
- With a **scripted interface** limited to just one user, namely the owner of the FlowTracer project. This mode is similar to the one used to interface to LSF or SGE. The advantage of this approach is that the same `nc run` interface is used to submit the jobs, so all jobs are processed by all policy layer implemented as part of `nc run`.

Using the Binary Interface

FlowTracer provides an optimized implementation in C++ for the interface to Accelerator.

As discussed in [Connect to a Third Party Batch Processing System](#), a Batch Processing System agent (BPS agent) is a `vovtasker` that connects to your FlowTracer `vovserver` and dispatches jobs to a backend Batch Processing System. You can start a BPS agent `vovtasker` from the command line by using the `-V` option of the `vovtasker` command, or you can define a BPS agent to Accelerator in `taskers.tcl` so that the BPS agent will be started when you start the FlowTracer project.

Start a BPS Agent from Command Line

First, decide which Accelerator to use to send jobs to which Accelerator project. Usually the Accelerator queue is called 'vnc', but you can find the available ones by asking your administrator, or by using:

```
% vovproject list -all -l | grep vnc
```

The command line syntax to start a BPS agent is:

```
% vovtasker -V qname@host[:port]
```

where `port` is optional and can be omitted if the [port number](#) of the Accelerator server is the default one for the queue name.

For example, an Accelerator server is running with the default name `vnc`, and is running on host `alpaca` with the default port `6271`. You can connect a BPS agent to the FlowTracer project and send jobs to this Accelerator server by:

```
% vovtasker -V vnc@alpaca          # or  
% vovtasker -V vnc@alpaca:6271
```

You can use other options of `vovtasker` to specify other [attributes](#) of your BPS agent. Among all the attributes, the default value of "capacity" for BPS agent is 100 while default for normal tasker is equal to the number of tasker machine's CPUs. You can override this default using the `-T` option. Usually you want to have a large capacity, because you want to give the BPS the ability to schedule jobs in the most effective manner. Capacities of 200 or 300 are typical.

Define a BPS Agent in taskers.tcl

In the `taskers.tcl` configuration file, you use `vtk_tasker_nc` to define a BPS agent to Accelerator.

After you define the BPS agent in `taskers.tcl`, it will be started when you restart your FlowTracer project. You can start it without restarting your project by using the command

```
% vovtaskermgr start name-of-BPS-agent
```

Unless you give it a different name, the name of the BPS agent will be `NC`. See instructions for [The taskers.tcl File](#) for details.

Examples

Here is an example, where in the context of FlowTracer project `test1@alpaca`, we now start a BPS agent which interfaces to FlowTracer `vnc@alpaca`:

```
% vovtasker -V vnc@alpaca -a NC -r "se_license spice_license"
```

Here's what the tasker monitors look like:



ft VOV Monitors 1212.1.0-dev-1212 denby_vov_main			
File Options TaskersGroups Taskers Resources			
TaskerGroups	Taskers	Tasker HW	Tasker Resources
1	localhost	READY	Unlim. localhost

Figure 8:

Here is the architecture diagram in this case:

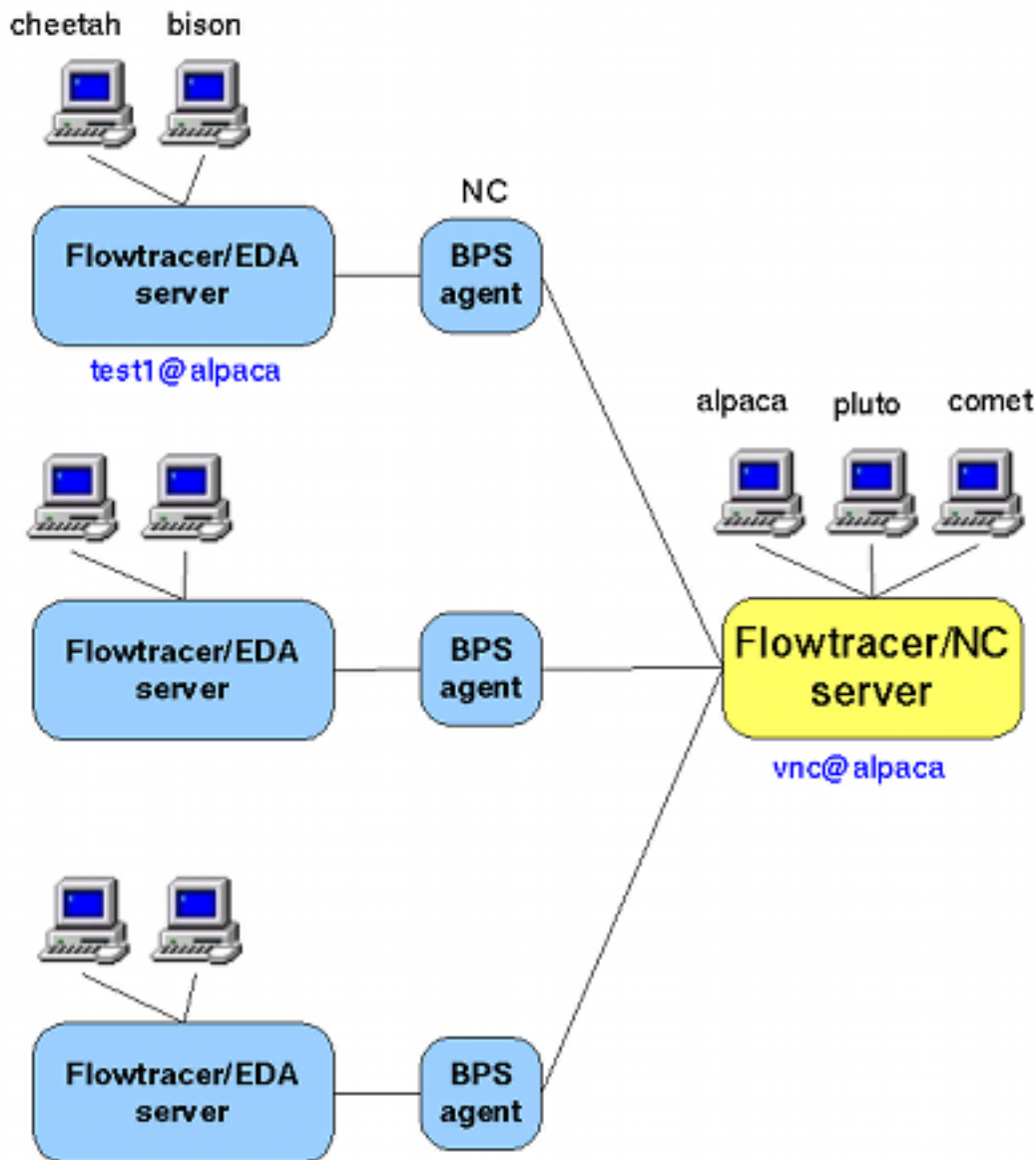


Figure 9:

The above example is a FlowTracer project named `test1@alpaca`. There are two normal "direct" taskers `cheetah` and `bison` already connected to it. The Accelerator server `vnc@alpaca` has 3 taskers, `alpaca`, `pluto` and `comet`. A BPS agent named `NC` is started in the context of `test1@alpaca`, which connects it to Accelerator `vnc@alpaca`. The taskers of `vnc@alpaca` are represented as BPS taskers prefixed with the name of the BPS agent in `test1@alpaca` for easy monitoring and management.

Handling Resources

The server managing the FlowTracer project uses its resource maps to determine which vovtasker should receive each job. When a job requests a floating resource, for example a license, that request needs to be passed through the BPS agent to Accelerator, so the job will not be started when the license is unavailable.

There are several methods you can use so that the jobs will run correctly in Accelerator and be dispatched through the BPS agent.

- If the number of resources is small, you can declare them using the `-resources` option of `vtk_tasker_nc`, or the `-r` option with the `-V` option of `vovtasker` on its command line.
- When the jobs in your FlowTracer project need many different resources from Accelerator, you can use the `ncconfig.tcl` file with the `-F` option of `vovtasker` for the BPS agent.

You can start the BPS agent `vovtasker`

- with the `vovtaskermgr` command
- automatically with a `liveness` script
- manually from the command line

Here is an example of the interface configuration file between FlowTracer and Accelerator.

```
# Interface control file for NC dispatcher (-V) vovtasker for an FT project
#pararray NCTASKER; # uncomment for troubleshooting

# Integer, controls debug messages
set NCTASKER(flag,debug) 0

# int, whether to show the farm tasker LEDs in the FT GUI
set NCTASKER(flag,show,taskers) 0

# BPS agent offers resources of this type from NC, e.g License:PrimeTime-SI
set NCTASKER(tasker,resources,resmap,types,export) {License}
# set NCTASKER(tasker,resources,resmap,types,export) {Priority}

# These are offered by the BPS agent in addition to the exported res from NC
set NCTASKER(tasker,resources) "NC linux sun7"

# set NCTASKER(job,resources,start) ""

# Resources to delete from jobs dispatched to NC
# set NCTASKER(job,resources,del) ""

# Resources to add to jobs dispatched to NC
# set NCTASKER(job,resources,add) ""


# set NCTASKER(job,group) ""
```

You can place this file in any convenient directory, but it is recommended to place it in the `.swd` of the FlowTracer project. This directory must be accessible to the FlowTracer project anyway, and the `live_start_nctasker.tcl` script expects it to be there.

When using the interface file with BPS taskers defined in `taskers.tcl` there is a shortcut that causes the file to be searched for first in the `.swd` of the FT project, and then in `$VOVDIR/local`.

To setup your FlowTracer project so that an Accelerator BPS agent can be managed with the `vovtaskermgr` command, use the `-nccfg` option of `vtk_tasker_nc`, for example:

```
vtk_tasker_nc -name NCbps -nccfg path-to-ncconfig.tcl
```

 **Note:** If you use the special value - (minus sign) for `path-to-nconfig.tcl`, the search mechanism described above will be invoked. Whether by search or path specification, the config file must be found or the BPS vovtasker will not be started.

To manually start a BPS agent vovtasker from the command line with the interface configuration file, use:

```
% vovtasker -V qname@host[:port] -F path-to-nconfig.tcl
```

To setup your FlowTracer project so that an Accelerator BPS agent will start automatically whenever jobs need to be run in your FlowTracer project, set up the `nconfig.tcl` file in your FlowTracer project's `.swd` directory. This is the only place where the standard liveness script looks for the interface file.

Also, copy the example liveness script to the `tasks` subdirectory of your FlowTracer project's `.swd` configuration directory. You may need to make the `tasks` directory there.

```
% cp $VOVDIR/etc/liveness/live_start_nctasker.tcl `vovserverdir -p tasks`
```

To test whether it is working, retrace some jobs from your FlowTracer project. You may need to modify the jobs so that they request the resource 'NC' that is offered by the BPS dispatcher. They should become SCHEDULED (light blue color). A new vovtasker should appear, then the job will become dispatched (cream color) and then RUNNING (yellow), and finally VALID or FAILED.

Check the resources offered by the BPS dispatcher tasker using the monitors, the browser, or by

```
% vovtaskermgr show -resources
```

The log file for the BPS dispatcher is in the usual place for tasker logs, in the directory `.swd/logs/taskers/the-tasker-name`.

Using the Scripted Interface

These taskers can be declared in the same way as the binary interface, except that you have to use the option `-singleUser 1` in `vtk_tasker_nc`. The script `$VOVDIR/etc/tasker_scripts/taskersVNC.tcl` provides an interface to Accelerator.

Interfaces to Other Batch Processing Systems

You can find examples of interfaces in the directory `$VOVDIR/etc/tasker_scripts`. The file `$VOVDIR/etc/tasker_scripts/taskerLSF.tcl` implements the interface to LSF.

Use Implemented Interface to SGE

You can use the `-I` option of `vovtasker` to start a BPS agent. For example, FlowTracer implements the interface to SGE in `$VOVDIR/etc/tasker_scripts/taskerSGE.tcl`. You can start the agent to SGE like this:

```
% vovtasker -I $VOVDIR/etc/tasker_scripts/taskerSGE.tcl -T 300 -r "dc_shell_license  
dracula_license"
```

You can use other options of `vovtasker` to specify other [attributes](#) of your BPS agent. Among all the attributes, the default value of "capacity" for BPS agent is set to 100 (while the default for a normal tasker is equal to the number of tasker machine's CPUs). You can always overwrite this default by -T option. Usually you want to have a large capacity, because you want to give the BPS the ability to schedule jobs in the most effective manner. Capacities of 200 or 300 are typical.

Study and Write Your Own Implementation to Other BPS

The BPS agent is implemented in a BPS agent file, which is a Tcl script that implements the procedures described in the table below.

To implement the BPS agent for your BPS system, for example, LSF, you just need to implement the procedures in the following table. You are encouraged to use the provided implemented interface to SGE as example.

Procedure	Description
<code>taskerStartJob</code>	This procedure submits a job to the BPS. All information about the job is in the global array <code>jobDesc</code> . It should return a reference id, i.e. the id of the job in the context of the BPS.
<code>taskerStopJob</code>	This procedure is called when the tasker wants to stop a running or queued job.
<code>taskerCheckJob</code>	This procedure is called when the tasker wants to check the status of a job. It returns one of the following values: LOST DONE FAILED RUNNING QUEUED.
<code>taskerResumeJob</code>	This procedure is called when the tasker wants to resumed a suspended job.
<code>taskerSuspendJob</code>	This procedure is called when the tasker wants to suspend a job.
<code>taskerJobEnded</code>	This procedure is called when the tasker receives a notification from the server that a job has ended. This allows the tasker to update its state promptly instead of waiting for another timeout.
<code>taskerMapResources</code>	A procedure to map the resources required by the job in the context of the local project and the resources required by the job in the context of the BPS.
<code>taskerCleanup</code>	This procedure is called when the indirect tasker exits. It should be used to cleanup the garbage that may have been created by the tasker.

`taskerStartJob`

The procedure `tasker Start` takes a single argument, the `jobId`. The rest of the job information is available in the array `jobDesc`, as described in the following table:

Variable Name	Meaning
<code>jobDesc (command)</code>	The complete command line.
<code>jobDesc (env)</code>	The environment label for the job.
<code>jobDesc (id)</code>	The job Id.
<code>jobDesc (priority)</code>	The VOV priority level.
<code>jobDesc (resources)</code>	The resource list for the job.
<code>jobDesc (user)</code>	The user that owns the job.
<code>jobDesc (xdur)</code>	The expected duration.

The procedure returns the reference ID used by the BPS. Example:

```
proc taskerStartJob { jobId } {
  global jobDesc env
  # Generate a label from the command by eliminating
  # all non alphanumeric characters.
  set label $jobDesc(command)
  regsub -all {[^a-zA-Z0-9_]+} $label "_" label
  set label [string range $label 0 7]
  set submitInfo [exec qsub -V -v VOV_ENV=BASE -j y -N $label $env(VOVDIR)/scripts/
vovfire $jobId]
  set refId [lindex $submitInfo 2]
  return $refId
}
```

taskerStopJob

This procedure is called when the tasker wants to stop a job. The procedure takes two arguments: the VovId of the job and the referenceId returned by taskerStartJob.

Example:

```
proc taskerStopJob { jobId refId } {
  # Stop a SGE job.
  exec qdel $refId
}
```

taskerCheckJob

This procedure takes two arguments: the VovId of the job and the referenceId returned by taskerStart. It is called when the tasker wants to find out the status of a job. The procedure is expected to return one of the following values:

Return Value	Meaning
LOST	The job is no longer in the BPS. It is generally assumed that the job is done.
DONE	The job is done.

Return Value	Meaning
FAILED	The BPS believes that the job has failed.
RUNNING	The job is currently executing.
QUEUED	The job is in the BPS queue.

Example:

```
proc taskerCheckJob { jobId refId } {
    # Check status of a SGE job.
    set status [ParseOutputOf [exec qstat] $refId]
    if { $status == "RUNNING" } {
        vtk_tasker_job_started $jobId [GetStartTime $refId]
    }
    return $status
}
```

If the job is running, it is the responsibility of this routine to inform the tasker of the exact time the job was started by invoking:

```
vtk_tasker_job_started $jobId $timespec
```

taskerSuspendJob, taskerResumeJob

These procedures are used to suspend and resume a job. These procedures also take two parameters: the *jobId* and the *referenceId*.

taskerJobEnded

This procedure is called when the tasker receives a notification from the server that a job has ended. This allows the tasker to update its state promptly instead of waiting for another timeout.

taskerCleanup

This procedure is called when the indirect tasker exits. It should be used to cleanup the garbage that may have been created by the tasker.

Configuration File for vovagent

For cases where the vovtaskers are started by submitting the binary to a separate batch queue system such as LSF or SGE, the system administrator may decide to use *vovagent*, which is a special copy of the *vovtasker* binary.


When *vovagent* is invoked, the binary limits the values of attributes that are based on the information stored in the configuration file `$VOVDIR/local/vovagent.cfg` to ensure compliance with the underlying batch queue system's FairShare policies.

The contents of the configuration file and relation to the command line and *taskers.tcl* file parameters are as follows:

Attribute Name	vovtasker options	vovagent.cfg option	Description
<i>maxidle</i>	-Z	<i>maxIdleTime</i>	After being idle for the given time, tasker does not accept new jobs and exits after completing active jobs. Value is a VOV timespec, for example, 2m. The minimum value is 10s, and the maximum is the value of the <i>maxlife</i> parameter or 1hour, whichever is less. If the configuration file is absent or does not specify a value, the default is 1m.
<i>maxlife</i>	-Z	maxLifeTime	After the specified lifetime, tasker does not accept new jobs and exits after completing active jobs. The value is a VOV timespec, for example, 2H. The minimum value is 5m, and the maximum of unlimited is specified by giving 0 (zero) or a negative value. If the configuration file is absent or does not specify a value, the default is 2H.
<i>update</i>	-U	<i>update</i>	Specifies the update cycle time for the vovtasker agent, which is the interval at which tasker resource procedures are recalculated. The minimum value is 5s, and the maximum is half the value of the <i>maxidle</i> parameter or

Attribute Name	vovtasker options	vovagent.cfg option	Description
			5 minutes, whichever is less. If the configuration file is absent or does not specify a value, the default is 1m.

An example of the `vovagent` configuration file is shown below.

 **Note:** Set access permissions to ensure that only the VOV system manager account can modify this file.

```
maxIdleTime = 1m
maxLifeTime = 1h
updateInterval = 15s
```

Subordinate Resources

The subordinate resources are the resources of a job that appear after the first '--' in the resource specification.

For example, in the resource specification:

```
JobType:compile -- ( alpaca OR sun7 )
```

The subordinate resources is the string `(alpaca OR sun)`

These subordinate resources are used by indirect taskers in the submission of the job to the remote BPS.

In the following example, the primary resource is `NC`, which is a simple way to route a job towards an indirect tasker connected to Accelerator, while `License:drc1 OR License:drc2` is the subordinate resource which will be used when the job is transferred to Accelerator.

```
R "NC -- License:drc1 OR License:drc2"
J vw run_some_drc Chip.x
```

The field to access the subordinate resources is called "SUBRESOURCES".

Interface with LSF using taskerLSF.tcl

Start the LSF Tasker

You can start the tasker from the command line:

```
% vovtasker -a LSF -i 0 -I $VOVDIR/etc/tasker_scripts/taskerLSF.tcl -r "lsf" -T 100
```

Alternatively, you can define the tasker in the taskers.tcl file:

```
# Fragment of taskers.tcl file
vtk_tasker_define localhost -name LSF -indirect $env(VOVDIR) /
etc/tasker_scripts/taskerLSF.tcl -transient 0
-resources "lsf" -capacity 100
```

The tasker is **non transient**, to preserve the status of the jobs submitted to LSF even in the event of a crash of the tasker.

Environment Variable VOV_LSF_QUEUES

The variable VOV_LSF_QUEUES is a comma separated list of queue names. If the variable is defined, the first queue in the list is going to be the default queue to which the jobs are submitted. This default can be overridden on a job by job basis by defining the resource "Queue:QNAME".

Job Submission with vovfire

Each job submitted from taskerLSF.tcl uses vovfire. The submission command is going to have the following form:

```
bsub [BSUB OPTIONS] vovfire 00012345 label_computed_from_command_line
```

Resource Mapping

The procedure taskerMapResources in file taskerLSF.tcl performs mapping of job resources into LSF submission specifications. This procedure can be redefined in a file called \$VOVDIR/local/taskerLSF.tcl. The default mapping is described in the following table.

Job Resource	LSF bsub parameter	Example	
Jobname:jname	-J jname	Jobname:abc	-J abc
LSFcpus#N	-n \$N	LSFcpus#4	-n 4
LSFtiles#N	-n \$N	LSFtiles#2	-n 2
License:lic	rusage [lic=1]	License:qx	rusage [qx=1]
Queue:qname	-q qname	Queue:night	-p night
RAM/*	rusage [mem=*]	RAM/200	rusage [mem=200]
RAMFREE/*	rusage [mem=*]	RAMFREE/200	rusage [mem=200]

Job Resource	LSF bsub parameter	Example	
swp/*	rusage[swp=*]	swp/100	rusage [swp=100]
tmp/*	rusage[tmp=*]	tmp/200	rusage [tmp=100]

anything else is ignored

Interface to LSF with vovlsfd

The daemon `vovlsfd` enables Accelerator or FlowTracer to allocate jobs on CPUs managed by LSF while avoiding the taxing per-job scheduling overhead imposed by LSF.

Jobs that request an LSF queue resource containing the prefix "LSFqueue:" are candidates to be dispatched by `vovlsfd`.

The basic idea is that, on demand, `vovlsfd` submits to LSF a request to execute `vovtaskerroot`. Once `vovtaskerroot` connects, Accelerator or FlowTracer can use it to execute one or more jobs that request a LSFqueue resource.

Configure the vovlsf Wrapper

`vovlsfd` makes use of the `vovlsfd` wrapper script that is included as part of the installation. The configuration file for `vovlsfd` is located in `$VOVDIR/local/vovlsf.config.csh`. If the file does not exist, create it and populate it with information to configure a shell to use your site's particular LSF setup. Once setup, the following commands should work in with a `vovproject` enabled terminal.

```
% vovlsf bsub sleep 10
Job <37655> is submitted to default queue <normal>.
% vovlsf bjobs
JOBID   USER      STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
37655   jeremym  PEND   normal     buffalo     sleep 10    Mar  6 11:56
% vovlsf bqqueues
QUEUE_NAME  PRIO STATUS      MAX JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
normal      30   Open:Active  -   -   -   -       1     0    1    0
% vovlsf bkill 37655
Job <37655> is being terminated
```

Configure vovlsfd

In order to use `vovlsfd`, the configuration file `PROJECT.swd/vovlsfd/config.tcl` must exist.

You can start from the example shown here, by copying it from `$VOVDIR/etc/config/vovlsfd/config.tcl`.

```
# Example Configuration file for vovlsfd:
# -----
# Remember to add the resource "LSFqueue:$myqueue" to your LSF jobs,
# or vovlsfd will assume the jobs are for direct/already connected taskers.

# How often should the vovlsfd daemon cycle?
```

```
# The value is a VOV time spec and the default is two seconds.
set VOVLSDSFD(refresh) 2s

# how frequently should we ask bjobs for status?
# The value is a VOV time spec and the default value is one minute.
set VOVLSDSFD(bjobs,checkfreq) 1m

# How often should we check for sick taskers?
# The value is a VOV time spec and the default is one minute
set VOVLSDSFD(sick,checkfreq) 1m

# Remove sick taskers that are older than?
# Value is a VOV time spec, and the default value is five minutes.
set VOVLSDSFD(sick,older) 5m

# Should we dequeue any extra taskers?
# Setting this to "1" will cause a dequeue
# of all not yet running vovtasker submissions for a job bucket.
# This only happens after three consecutive refresh cycles
# have gone by with no work scheduled for that bucket.
set VOVLSDSFD(dequeueExtraTaskersEnable) 1

# How long should we wait to dequeue any extra taskers?
# The number of refresh cycles
set VOVLSDSFD(dequeueExtraTaskersDelay) 3

# What is the maximum number taskers we should start?
# Should be set to a high value to enable lots of parallelism.
set VOVLSDSFD(tasker,max) 99

# What is the maximum number of queued taskers per bucket that we should allow?
set VOVLSDSFD(tasker,maxQueuedPerBucket) 99

# How many tasker submissions should be done for each
# job bucket, during each refresh cycle?
set VOVLSDSFD(tasker,maxSubsPerBucket) 1

# What is the minimum number of taskers that will be grouped into an array
# for each resource bucket, during each refresh cycle?
# Setting this to "0" will disable LSF array functionality.
set VOVLSDSFD(tasker,lsfArrayMin) 0

# What is the maximum number of taskers that will be grouped into an array
# for each resource bucket, during each refresh cycle? The absolute max number
# of taskers supercedes this value.
# Setting this to "0" will disable LSF array functionality.
set VOVLSDSFD(tasker,lsfArrayMax) 0

# What is the longest a vovtasker should run before self-exiting?
# Ex: if you set it to 8 hours, and queue 4 3-hour jobs:
# the first tasker will run for nine hours (3 x 3-hr > 8-hr) and then exit
# the fourth job will only start when a second tasker has been requeued
# and started by the batch execution system.
# This controls the amount of reuse of a tasker while it processes jobs.
# To avoid the penalties of:
# noticing a tasker is needed
# + submitting to the batch system
# + the batch system to allocate a machine
# You should set this to a high value like a week.
# The value is a VOV time spec
# This is a default value. It can be overridden on a per job basis by putting
# a resource on the job that looks similar to the following.
# MAXlife:1w
```

```
set VOVLSD(tasker,maxlife)      1w

# How long should a tasker wait idle for a job to arrive?
# The shorter time, the faster the slot is released to the batch system.
# The longer time, the more chances the tasker will be reused.
# The default value is two minutes (usually takes a minute to allocate a
# slot through a batch system). Value is a VOV time spec
# This is a default value. It can be overridden on a per job basis by putting
# a resource on the job that looks similar to the following.
# MAXidle:2m
set VOVLSD(tasker,maxidle)      2m

# Are there any extra resources you wish to pass along to the taskers?
# These resources will be passed directly along to the vovtasker. They
# are not processed in any way by vovlsfd. For example setting
# this to "MAXlife:1w" will not work as you might expect.
set VOVLSD(tasker,res)          ""

# Is there a default string you wish to pass as "-P" on the bsub
# command line. Per job settings are also supported by putting a
# resource on the job that looks similar to the following.
# LSFptag:blah
set VOVLSD(tasker,ptag)         ""

# What is the vovtasker update interval for resource calculation?
# Value is a VOV time spec, and the default value is 15 seconds.
set VOVLSD(tasker,update)       15s

# How many MB of RAM should we request by default?
# (Default: 256MB)
set VOVLSD(tasker,ram)          256

# How many CPUs on the same machine should we request by default?
# (Default: 1 core)
set VOVLSD(tasker,cpus)         1

# What is the default LSF jobname to be used when launching vovtaskers?
set VOVLSD(tasker,jobname)       "ft_${env(VOV_PROJECT_NAME)}"

# Do we want to enable debug messages in the vovtasker log files?
# 0=no; 1=yes; default=0
set VOVLSD(tasker,debug)        0

# What level of verbosity should the vovtasker use when writing to its the log file?
# Valid values are 0-4; default=1
set VOVLSD(tasker,verbose)      1

# How long should the vovtasker try to establish the initial connection to the
# vovserver?
# Values are in seconds, default is 120 seconds.
set VOVLSD(tasker,timeout)      120

# How many hosts should the vovtasker span?
# The value can be overridden by an individual job using the subordinate resource
# functionality.
# Without that override, this config variable defines a default value for
# span[hosts=*]
# Setting the value to zero omits the default span[hosts=*] and allows the underlying
# batch
# scheduler to use its own internally set value.
# Setting to -1 disables the span setting defined for the queue in the batch
# scheduler configuration.
# Setting to 1 tells the batch system to provide all processors from a single host.
```

```
# https://www.ibm.com/support/knowledgecenter/SSETD4_9.1.3/lsf_admin/span_string.html
# Valid values are -1, 0, and 1; default=1
set VOVLsFD(tasker,spanHosts)      1

# Which precmd script should we include in the bsub command line by default?
# The precmd script must be an executable script located in the
# VOVLsFD(launchers.dirname) directory.
# The script should be self contained so that we can reference it with a single word.
# (i.e. no arguments, and not full paths)
set VOVLsFD(tasker,precmd)         ""

# How much buffer should we consider when adjusting tasker,max based on available
# client connections?
# This number will be subtracted from the available maxNormalClient connections
# Twice this number will be subtracted from the available file descriptors
# Set this number based on how many non-vovtasker client connections are anticipated
# for this session.
set VOVLsFD(client,derate)         50

# What is the name of the launchers directory? (Default: \"../launchers\")
set VOVLsFD(launchers,dirname)     \"./launchers\"

# How often should we check the launchers directory for a cleanup?
# The value is a VOV time spec and the default is 10 minutes
set VOVLsFD(launchers,checkfreq)  10m

# Remove launchers that are older than?
# Value is a VOV time spec, and the default value is one hour.
set VOVLsFD(launchers,older)       1h

# Generate a CSV report file (pendingrunning.csv) upon each pass
# of the daemon cycle.
set VOVLsFD(savePendingRunningReport) 0

# queue-specific options:
# set VOVLsFD(bsub,options,$qName)

## To control the name of the executable that is dispatched to LSF.
## The -sr flag in vov_lsf_agent is what
# set VOVLsFD(exeleaf) \"$env (VOVDIR) /scripts/vovboot vov_lsf_agent\"
# set VOVLsFD(exeleafargs) \"-sr\"
```

This file does not initially exist, so it will have to be manually created. Use the above example as a template. Here is a sequence of commands to set up vovlsfd for a given project.

```
% vovproject enable <project>
% cd `vovserverdir -p .`
% mkdir vovlsfd
% cp $VOVDIR/etc/config/vovlsfd/config.tcl vovlsfd/config.tcl
% vi vovlsfd/config.tcl ; # Edit config file to suit your installation.
```

Start vovlsfd Manually

Use vovdaemonmgr to start vovlsfd manually.

```
% vovproject enable <project>
% vovdaemonmgr start vovlsfd
```


Start vovlsfd Automatically

In the directory `vnc.swd/autostart`, create a script called `vovlsfd.csh` with the following content:

```
#!/bin/csh -f
vovdaemonmgr start vovlsfd
```

Don't forget to make the script executable.

```
% chmod 755 vovlsfd.csh
```

Start Multiple vovlsfd Daemons For Each User

While a single `vovlsfd` daemon is sufficient even when dealing with multiple users, it may be useful at times to have multiple `vovlsfd` daemons running for each user when running FlowTracer in a multi-user operating mode, mostly because it allows traceability of the user in LSF.

To enable the FlowTracer multi-user mode, set the environment variable `VOV_FT_MULTIUSER`. When `vovlsfd` is started with this environment variable set, each user can start his/her own instance of `vovlsfd`. And this instance will only process jobs for its owner user.

It is the responsibility of the user to keep his own `vovlsfd` daemon running properly. Each private `vovlsfd` daemon uses the config file from its working directory, which is under `<PROJECT>.swd/vovlsfd/<user>`. If the config file is not found in the user specific directory, `vovlsfd` will also check the directory above `<PROJECT>.swd/vovlsfd` for the config file.

Debug vovlsfd on the Command Line

When first starting `vovlsfd`, it is helpful to run it in foreground, possibly with the `-v` and/or `-d` options to verify operation as expected.

```
% vovproject enable <project>
% cd `vovserverdir -p vovlsfd`
% vovlsfd ; # Launch vovlsfd
```

Specify LSF Job Resources

To submit a job that is sent to be executed on a LSF CPU, assign the job the following resources:

Resource Name	Explanation	Notes
LSFqueue: <QUEUENAME>	This maps to <code>-q <QUEUENAME></code>	required
LSFlicense: <LSF_LICENSE>	This maps to a "rusage" spec of the type <code><LSF_LICENSE>=1:duration=1</code>	optional
LSFapp: <LSF_APP>	This maps to <code>-app <LSF_APP></code>	optional
LSFmopts: <LSF_MOPT>	This maps to <code>-m <LSF_MOPT></code>	optional
LSFjobname: <LSF_JOBNAME>	This maps to <code>-J <LSF_JOBNAME></code>	optional

Resource Name	Explanation	Notes
LSFptag:<LSF_PTAG>	This maps to -p <LSF_PTAG>	optional
LSFpre:<LSF_PRECMD>	<p>This maps to -E <LSF_PRECMD>. Its important that the value provided for <LSF_PRECMD> is a single word with no spaces or slashes. In other words no full paths or arguments. The script itself must be executable and exist in the <code>launchers/prescripts</code> directory. Typically, <code>SWD/vovlsfd/launchers/prescripts</code>.</p> <p>The LSF:pre special resource will use this directory as the base directory for specified prescript. Example: <code>LSFpre:mypre.sh</code> results in an LSF submission option of <code>-E ./prescripts/mypre.sh</code>.</p>	optional
Type:<LSF_TYPE>	This maps into a "select" statement of the form <code>type=<LSF_TYPE></code>	optional
RAM/<ram>	This maps into a "rusage" statement of the form <code>mem=<ram></code>	optional
CORES/<cpus>	This maps to -n <cpus>	optional
MAXlife:<TIMESPEC>	This overrides the default maxlife value given to the launched vovtasker. <TIMESPEC> is a VOV timespec.	optional
MAXidle:<TIMESPEC>	This overrides the default maxidle value given to the launched vovtasker. <TIMESPEC> is a VOV timespec.	optional
vovlsfd:<RESOURCE_NAME>	By declaring a resource token <code>vovlsfd:<RESOURCE_NAME></code> in <code>resources.tcl</code> with some finite limit, and using the resource on a job, vovlsfd will track	optional

Resource Name	Explanation	Notes
	the resource so as not to over submit jobs to LSF.	
TAG:<TAG>	This allows attributes to be passed to the resulting vovtasker, without being considered by the batch system. This allows the user to force jobs to execute only on vovtaskers that have the corresponding <TAG>.	optional

Submit Jobs to LSF using Accelerator

Examples of job submission with Accelerator:

```
% nc run -r LSFqueue:normal -- spice abc.spi
% nc run -r LSFqueue:night LSFresource:dc -- dc_shell -f script.tcl
```

Submit Jobs to LSF using FlowTracer (FDL)

Examples of assigning LSF resources to jobs with FlowTracer:

```
N "spice"
R "LSFqueue:normal"
J vw spice abc.spi

N "dc_shell"
R "LSFqueue:night LSFresource:dc"
J vw dc_shell -f script.tcl
```

It is also possible to pass additional LSF bsub options for a particular job to vovlsfd in two ways.

1. Applying the "BSUB_OPTIONS" text property to a job
2. Adding a specially formatted string (beginning with "--") to the end of the FDL 'R' procedure argument

It's up to you to determine which approach to use. It's a matter of preference.

```
# 1) Append subordinate string to resource string using the following format :
# R {<required resources> -- "<LSF bsub options>"}
N "md5"
R {LSFqueue:medium -- "-R 'select[type==64BIT]' -app 'highpri'"}
J vw md5.pl

# 2) or, apply a property to the job
N "md5"
R "LSFqueue:medium"
set jobId [J vw md5.pl]
vtk_prop_set -text $jobId BSUB_OPTIONS {-R 'select[type==64BIT]' -app 'highpri'}
```

Make note that text in Tcl square brackets will be interpreted as a Tcl command when contained within a double-quoted string, so to prevent any Tcl mischief, either square brackets must be back-slashed, or use curly brackets in place of double quotes.

Control the Values of vovlsfd Options

Because a normal user can change the `config.tcl` file and or change the resource settings for a job to modify LSF or vovtasker settings, it may be desirable to limit or police certain values to keep them in line with enterprise level policy. This can now be done by writing specially named Tcl procedures, and including those procedures in `$VOVDIR/local/vovlsfd/police.tcl`.

A police proc should be named `police_` concatenated with the array index name of the config variable in question. For example, if you want to write a proc that polices the `VOVLSFD(tasker,maxlife)` setting used to set the maximum life of a vovtasker submitted to LSF by `vovlsfd`, then the proc name is `police_tasker,maxlife`. The proc will be called with a single input, the current value of the variable. The proc should return the desired value.

```
# Example police.tcl proc that limits maxlife to five minutes or less.
proc police_tasker,maxlife { input } {
    if { [VovParseTimeSpec $input] > 300 } {
        return [vtk_time_pp 300]
    } else {
        return $input
    }
}
```



Note: Currently, only the `maxlife` and `maxidle` values in the `config.tcl` and the per job resource line are guaranteed to work with this new feature. Other variables can be added given demand and/or time for development.

Monitor LSF Jobs in the GUI

If the environment variable `VOV_LSF_QUEUES` is defined, the GUI monitor will show additional panels and menus to allow the monitoring of jobs dispatched to LSF using indirect taskers. This capability relies on a running `vovlsfinfod`.

To minimize the load on LSF, you can use an interface called `vovlsfinfod`, which is primarily the single entry for all requests by VOV scripts but also a handy HTTP server to visualize the status of the LSF system.

Start vovlsfinfod

This daemon needs to be started by the administrator user.

```
% cd $VOVDIR/../../..
% mkdir vovlsfinfod% cd vovlsfinfod
% touch config.tcl;          ### This file is to be empty for now.
% which bjobs
% vovlsfinfod
Sourcing default config file './config.tcl'
Configuration of vovlsfinfod:
```

```
=====
Running on host:      lnxlap
In directory:        /home/xxx/rtda/vovlsfinfod
Configuration file:   ./config.tcl
Refresh interval:    30
TCP/IP port:         6006
Log file directory:  ./logs
info file:           ./info.tcl
Verbosity level:     0
vovlsfinfod: message: You can now connect to http://HOSTNAME:6006
```

To monitor the LSF activity, point the browser to `http://HOSTNAME:6006` where HOSTNAME is the name of the host on which vovlsfinfod is running.

Compile vovbjobs

vovlsfinfod gets the job information using one of the following methods:

- Parses the output of "bjobs -l" to get the status of the LSF queues. This is difficult and slow.
- If available, it uses the utility vovbjobs. We only distribute the source code for vovbjobs in \$VOVDIR/src/vovbjobs. It is your responsibility to compile it into a binary.

Config File for vovlsfinfod

Use the config.tcl file to add links in the web pages to other sites using the following syntax:

```
# Example of config.tcl file for vovlsfinfod
# add_LINK LABEL URL
add_LINK  "FLEXlm"  "http://localhost:5005"
add_LINK  "FT/LM"   "http://localhost:6450/cgi/ftlm.cgi"
```

Interface with Altair Accelerator using vovelastcd

The daemon vovelastcd enables Altair Accelerator or Altair FlowTracer to allocate jobs on CPUs managed by Accelerator while avoiding per-job scheduling overhead.

Jobs that request an LSF queue resource containing the prefix "LSFqueue:" are ignored by vovelastcd. This allows vovlsfd and vovelastcd to cooperate and work together. All jobs that do not request an LSF queue are candidates for vovelastcd.

The basic idea is that, on demand, vovelastcd submits to Accelerator a request to execute vov_elastic_agent. Once vov_elastic_agent connects, Accelerator or FlowTracer can use it to execute one or more jobs.

In addition, the vovtasker connection timeout for elastic vovtaskers can be configured via the config.tcl file.

Configure vovelastcd

In order to use vovelastcd, the configuration file PROJECT.swd/vovelastcd/config.tcl must exist.

You can start from the example shown here, by copying it from \$VOVDIR/etc/config/vovelastcd/config.tcl.

```
# How often should the vovelastcd daemon cycle?
# The value is a VOV time spec and the default is two seconds.
set VOVELASTICD(refresh) 2s

# Which NC instance should the vovelastcd daemon submit jobs to?
set VOVELASTICD(queue) "vnc"

# Environment to be used when submitting an vov_elastic_agent.
set VOVELASTICD(runenv) "BASE"

# How old should a job bucket be before vovelastcd considers it for servicing?
# The value is a VOV time spec and the default is 5 seconds.
set VOVELASTICD(bucketAgeThreshold) 0

# What should we call jobs that are submitted to NC?
set VOVELASTICD(elasticJobName) "$daemonName"

# To which set in the NC instance should our jobs be appended?
set VOVELASTICD(elasticSetName) "$daemonName:$projInfo(name)@
$projInfo(host)@$projInfo(port)"

# How often should we check for failed job submissions?
# When a failed job submission is detected, it is removed
# from the internal data structures used for calculating
# whether or not to submit a new tasker.
set VOVELASTICD(failed,checkfreq) 1m

# How often should we check for sick taskers?
# The value is a VOV time spec and the default is one minute
set VOVELASTICD(sick,checkfreq) 1m

# Remove sick taskers that are older than?
# Value is a VOV time spec, and the default value is five minutes.
set VOVELASTICD(sick,older) 5m

# Should we dequeue any extra taskers?
# Setting this to "1" will cause a dequeue
# of all not yet running vovtasker submissions for a job bucket.
# This only happens after three consecutive refresh cycles
# have gone by with no work scheduled for that bucket.
set VOVELASTICD(dequeueExtraTaskersEnable) 0

# How long should we wait to dequeue any extra taskers?
# The number of refresh cycles
set VOVELASTICD(dequeueExtraTaskersDelay) 3

# What is the maximum number taskers we should start?
# Should be set to a high value to enable lots of parallelism.
set VOVELASTICD(tasker,max) 99

# What is the maximum number of queued taskers per bucket that we should allow?
set VOVELASTICD(tasker,maxQueuedPerBucket) 99

# How many tasker submissions should be done for each
# resource bucket, during each refresh cycle?
# Setting this to "0" will disable blitz functionality.
set VOVELASTICD(tasker,blitz) 0

# What is the minimum number of taskers that will be grouped into an array
```

```
# for each resource bucket, during each refresh cycle?
# Setting this to "0" will disable array functionality.
set VOVELASTICD(tasker,jobArrayMin) 0

# What is the maximum number of taskers that will be grouped into an array
# for each resource bucket, during each refresh cycle? The absolute max number
# of taskers supercedes this value.
# Setting this to "0" will disable array functionality.
set VOVELASTICD(tasker,jobArrayMax) 0

# What is the longest a vovtasker should run before self-exiting?
# Ex: if you set it to 8 hours, and queue 4 3-hour jobs:
# the first tasker will run for nine hours (3 x 3-hr > 8-hr) and then exit
# the fourth job will only start when a second tasker has been requeued
# and started by the batch execution system.
# This controls the amount of reuse of a tasker while it processes jobs.
# To avoid the penalties of:
# noticing a tasker is needed
# + submitting to the batch system
# + the batch system to allocate a machine
# You should set this to a high value like a week.
# The value is a VOV time spec
# This is a default value. It can be overridden on a per job basis by putting
# a resource on the job that looks similar to the following.
# MAXlife:1w
set VOVELASTICD(tasker,maxlife) 1w

# How long should a tasker wait idle for a job to arrive?
# The shorter time, the faster the slot is released to the batch system.
# The longer time, the more chances the tasker will be reused.
# The default value is two minutes (usually takes a minute to allocate a
# slot through a batch system). Value is a VOV time spec
# This is a default value. It can be overridden on a per job basis by putting
# a resource on the job that looks similar to the following.
# MAXidle:2m
set VOVELASTICD(tasker,maxidle) 2m

# Are there any extra resources you wish to pass along to the taskers?
# These resources will be passed directly along to the vovtasker. They
# are not processed in any way by vovelasticd. For example setting
# this to "MAXlife:1w" will not work as you might expect.
set VOVELASTICD(tasker,res) ""

# What is the vovtasker update interval for resource calculation?
# Value is a VOV time spec, and the default value is 15 seconds.
set VOVELASTICD(tasker,update) 15s

# Do we want to enable debug messages in the vovtasker log files?
# 0=no; 1=yes; default=0
set VOVELASTICD(tasker,debug) 0

# What level of verbosity should the vovtasker use when writing to its the log file?
# Valid values are 0-4; default=1
set VOVELASTICD(tasker,verbose) 1

# How long should the vovtasker try to establish the initial connection to the
# vovserver?
# Values are in seconds, default is 120 seconds.
set VOVELASTICD(tasker,timeout) 120

# How much buffer should we consider when adjusting tasker,max based on available
# client connections?
# This number will be subtracted from the available maxNormalClient connections
```

```
# Twice this number will be subtracted from the available file descriptors
# Set this number based on how many non-tasker client connections are anticipated for
  this session.
set VOVELASTICD(client,derate)          50

# What is the name of the launchers directory? (Default: \"./launchers\")
set VOVELASTICD(launchers,dirname)      \"./launchers\"

# How often should we check the launchers directory for a cleanup?
# The value is a VOV time spec and the default is 10 minutes
set VOVELASTICD(launchers,checkfreq)    10m

# Remove launchers that are older than?
# Value is a VOV time spec, and the default value is one hour.
set VOVELASTICD(launchers,older)        1h

# How often should we check for preempted jobs?
# Value is a VOV time spec, and the default value is one minute.
set VOVELASTICD(preempt,checkfreq)      1m

# What action should we take on a preempted vovtasker?
# Allowable values are STOP and RESERVE.
# STOP is faster than RESERVE but requires > 2013.09u5
set VOVELASTICD(preempt,taskerstop)     \"STOP\"

# Should we allow vovtasker to be preempted?
# Value is 1 or 0.
# The preemption request must come from an appropriately configured NC.
set VOVELASTICD(preemptTaskersEnable)    0

# In FT 2013.09u6 and possibly later, this setting should be 1,
# if the underlying batch system is NC and NUMA support is required by the taskers.
set VOVELASTICD(resources,numamap)      1

# The maximum number of launcher attempts that should be made in the event the job
  submission
# to the back-end queue fails. In most cases, this would be due to a misconfigured
  queue name.
# The counter is reset once the configuration file has been updated.
set VOVELASTICD(maxQueueErrors)          10
```

This file does not initially exist, so it will have to be manually created. Use the above example as a template. Here is a sequence of commands to set up vovelasticd for a given project.

```
% vovproject enable <project>
% cd `vovserverdir -p .`
% mkdir vovelasticd
% cp $VOVDIR/etc/config/vovelasticd/config.tcl vovelasticd/config.tcl
% vi vovelasticd/config.tcl ; # Edit config file to suit your installation.
```

Start vovelasticd Manually

Use vovdaemonmgr to start vovelasticd manually.

```
% vovproject enable <project>
% vovdaemonmgr start vovelasticd
```


Start vovelasticd Automatically

In the directory `vnc.swd/autostart` create a script called `vovelasticd.csh` with the following content:

```
#!/bin/csh -f
vovdaemonmgr start vovelasticd
```

Don't forget to make the script executable.

```
% chmod 755 vovelasticd.csh
```

Debug vovelasticd on the Command Line

When first starting `vovelasticd`, it is helpful to run it in foreground, possibly with the `-v` and/or `-d` options to verify operation as expected.

```
% vovproject enable <project>
% cd `vovserverdir -p vovelasticd`
% vovelasticd ; # Launch vovelasticd
```

Specify Job Resources

To submit a job that is sent to be executed on an Accelerator CPU, assign the job resources as you normally would for Accelerator. The resource request is passed along to Accelerator without modification. The following resources require special handling by `vovelasticd` internally.

Resource Name	Explanation	Notes
MAXlife:<TIME>	This overrides the default maxlife value given to the launched vovtasker. \$TIME is a VOV timespec.	optional
MAXidle:<TIME>	This overrides the default maxidle value given to the launched vovtasker. \$TIME is a VOV timespec.	optional
vovelasticd:<RESOURCE_NAME>	By declaring a resource token <code>vovelasticd:\$RESOURCE_NAME</code> in <code>resources.tcl</code> with some finite limit, and using the resource on a job, <code>vovelasticd</code> will track the resource so as not to over submit jobs to the batch queuing system.	optional
TAG:<TAG>	This allows attributes to be passed to the resulting vovtasker, without being considered by the batch system.	optional

Resource Name	Explanation	Notes
	This allows the user to force jobs to execute only on vovtaskers that have the corresponding <TAG>.	

Submit Jobs to Accelerator using Accelerator

Examples of job submission with Accelerator:

```
% nc run -r MAXlife:1h -- spice abc.spi
% nc run -r CORES/2 MAXidle:1m -- dc_shell -f script.tcl
% nc run -clearcaseResource -r CORES/2 vovelasticd:dcThrottle/2 -- dc_shell -f
script.tcl
```

Submit Jobs to Accelerator using FlowTracer (FDL)

Examples of assigning resources to jobs with FlowTracer:

```
N "spice"
R "MAXlife:1h"
J vw spice abc.spi


N "dc_shell"
R "CORES/2 MAXidle:1m"
J vw dc_shell -f script.tcl
```

Advanced Tasker Topics

Hardware Resources

All taskers offer a predefined set of hardware resources that can be requested by jobs.

All taskers offer a predefined set of hardware resources that can be requested by jobs. These resources are listed in the following table.

Hardware Resource	Type	Description
ARCH	STRING	The VOV architecture of the machine, for example "linux64", "win64", "armv8"
CORES	INTEGER	Consumable resource: the number of logical CPUs/processors used by a job.
CORESUSED	INTEGER	The total number of cores used by the running jobs. It is assumed that each job uses at least one core.
CLOCK	INTEGER	The CPU-clock of at least one of the CPUs on the machine in MHz. If the machine allows frequency stepping, this number can be smaller than expected.
GPUS	INTEGER	<div>  Note: The environment variable "VOV_GPU_SUPPORT_ENABLE" must be set in the tasker environment to enable GPUS support. The value of the environment variable is ignored. </div> <p>This will consume the specified number of GPUS resources and, in addition, the associated GPUs' GPU:<UUID>, GPU:<Model_Name> and GPU:RAM resources.</p> <p>Example:</p> <pre>nc run -v 1 -r GPUS/1 - sleep 10</pre>
GPU: <uuid>	INTEGER	Request GPUs by device name (UUID). This will consume one GPUS resource, and in addition, the associated GPUs' GPU:<UUID>, GPU: <Model_Name> and GPU:RAM resources.

Hardware Resource	Type	Description
		<p>Example:</p> <pre>nc run -v 1 -r GPU:GPU-ab3207e3-6dff-fcbe-d93b-0f91cb2d45c3/1 --sleep 10</pre>
GPU: <model_name>	INTEGER	<p>Request GPUS(s) by GPU model name. This will consume the specified number of GPU:<Model_Name> resources and, in addition, the specified number of GPUS resources, the corresponding GPU:<UUID> resources, and the corresponding GPU:RAM resource quantity.</p> <p>Example:</p> <pre>nc run -v 1 -r GPU:Quadro_K3100M/1 -- sleep 10</pre>
GPU:RAM	INTEGER	<p>This will request a GPU and the requested GPU:RAM from it, 1 GPUS resource and, in addition, the associated GPUs' GPU:<UUID> and GPU:<Model Name> resources.</p> <p>Example:</p> <pre>nc run -v 1 -r GPU:RAM/1024 - sleep 10</pre> <p>Note that even though the entire GPU:RAM might not be requested, the entire GPUS is consumed.</p>
GROUP	STRING	The tasker group for this tasker. Each tasker can belong to only one tasker group.
HOST	STRING	The name of the host on which the tasker is running. Typically this is the value you get with <code>uname -n</code> , except only the first component is taken and converted to lowercase, so that if <code>uname -n</code> returns <code>Lnx0123.my.company.com</code> the value of this field will be <code>lnx0123</code> .
LOADEFF	REAL	The effective load on the machine, including the self-induced load caused by jobs that just started or finished.
L1	REAL	On UNIX, the load average in the last one minute.
L5	REAL	On UNIX, the load average in the last five minutes.
L15	REAL	On UNIX, the load average in the last fifteen minutes.
MACHINE	STRING	Typically the output of <code>uname -m</code> .
MAXNUMACORES	INTEGER	Highest total number of NUMA cores in a single NUMA node.

Hardware Resource	Type	Description
MAXNUMACORESFREE	INTEGER	Highest number of free cores in a single NUMA node. Note that free NUMA cores are correctly accounted for only if the user specified -jpp pack or -jpp spread for all jobs on the tasker.
NAME	STRING	The name of the tasker.
OS	STRING	The name of the operating system: "Linux" or "Windows".
OSCLASS	STRING	This can be <code>unix</code> or <code>windows</code> .
OSVERSION	STRING	The version of the OS. On Linux, this can usually be found in <code>/etc/system-release</code> .
OSRELEASE	STRING	Typically the output of <code>uname -r</code> .
PERCENT	INTEGER	Consumable resource: The percentage of the machine that is still available.
POWER	INTEGER	The effective power of the tasker, after accounting for both raw power and the effective load.
RAM	INTEGER	A consumable resource expressing the remaining RAM available to run job: <code>RAMTOTAL-RAMUSED</code> , in MB.
RAMFREE	INTEGER	The amount of RAM available to run other jobs. This metric comes from the OS, and on linux it includes both free memory and buffers. In MB.
RAMTOTAL	INTEGER	The total amount of RAM available on the machine, in MB.
RAMUSED	INTEGER	The aggregate quantity of RAM used by all jobs currently running on the tasker, in MB. For each job, the amount of RAM is calculated as the maximum of the requested RAM resource (<code>REQRAM</code>) and the actual RAM usage of the job (<code>CURRAM</code>).
RELEASE	STRING	On Linux machines, this is the output of <code>lsb_release -isr</code> , with spaces replaced by dashes. For example, <code>CentOS-6.2</code>
SLOT	INTEGER	A consumable resource indicating how many more jobs can be run on the tasker.
SLOTS	INTEGER	Same as SLOT
SLOTSUSED	INTEGER	Corresponding to the number of jobs running on the tasker.

Hardware Resource	Type	Description
STATUS	ENUMERATED TYPE	Possible values are BLACKHOLE, BUSY, DEAD, DONE, FULL, OVRLD, NOLIC, NOSLOT OK, PAUSED, READY, REQUESTED, SICK, SUSP, WARN, WRKNG
SWAP	INTEGER	A consumable resource. The swap space in MB.
SWAPFREE	INTEGER	The amount of free swap.
SWAPTOTAL	INTEGER	Total about of swap configured on the machine.
TASKERNAME	STRING	Same as <i>NAME</i>
TASKERHOST	STRING	Same as <i>HOST</i>
TIMELEFT	INTEGER	The number of seconds before the tasker is expected to exit or to suspend. This value is always checked against the expected duration of a job.
TMP	INTEGER	On UNIX, free disk space in /tmp, in MB.
USER	STRING	The user who started the vovtasker server, which is usually the same user account associated with the vovserver process.
VOVVERSION	STRING	The version of the vovtasker binary (such as '2015.03').

Request Hardware Resources

Each job can request hardware resources.

 **Note:** The consumable resources are CORES, CPUS, PERCENT, RAM, SLOT, SLOTS, and SWAP.

- To request a machine with the name *bison*, request *NAME=bison*. To request any linux64 machine, request *ARCH=linux64*.
- Consumable resources are added together. For example *-r CORES/2 CORES/4 CORES/6* is a request for a total of 12 cores.
- If redundant resources are specified, the largest value will be taken. For example, if *-r RAMTOTAL#2000 RAMTOTAL#4000* is specified then *RAM#TOTAL4000* will be the resource that is used.

Request examples are listed in the following table:

Request Objective	Syntax for the Request
A specific tasker	<i>NAME=bison</i>
Not on bison	<i>NAME!=bison</i>

Request Objective	Syntax for the Request
One of two taskers	NAME=bison, cheetah
A preference: bison, if it is available; otherwise, cheetah	(NAME=bison OR NAME=cheetah)
A specific architecture, such as Linux	ARCH=linux
A specific tasker group, such as prodLnx	GROUP=prodLnx
2 GB of RAM	RAM/2000
Two cores	CORES/2
Two slots	SLOTS/2
Exclusive access to a machine	PERCENT/100
1 minute load less than 3.0	L1<3.0

HOST_OF_jobId and HOST_OF_ANCECEDENT

Sometime, a job needs to run on the same host as another job. This condition can be expressed with the special resource "HOST=HOST_OF_<jobid>" and "HOST=HOST_OF_ANCECEDENT".

If a job requires "HOST=HOST_OF_<jobid>", it will be run on the same host as the job with the specified id. If such job does not exist, the job will not run ever.

If a job requires "HOST=HOST_OF_ANCECEDENT", the scheduler looks for any of the jobs that generate at least one input of the job, and schedules the job on the same host. If the job has no antecedent, then the job will never run.

Examples of HOST= HOST_OF_ANCECEDENT

In the following example, both jobs are going to be executed on the same host.

```
J vw cp aa bb
R "HOST=HOST_OF_ANCECEDENT"
J vw cp bb cc
```

In the following example, the job j2 runs on the same host as job j1

```
R "RAM/10"
set j1 [J vw prepare_host_for_job]

R "RAM/1000 License:expensive HOST=HOST_OF_$j1"
set j2 [J vw run_big_job]
```

Tasker: vtk_tasker_set_timeleft

This procedure is used in the context of time-variant tasker resources. It takes a single non-negative argument, which is interpreted as the maximum expected duration for a job to be dispatched to the tasker.

For example, if a tasker does not want to accept jobs longer than 2 minutes, it will say:

```
namespace eval VovResources {
    proc SmallJobsOnly {} {
        vtk_tasker_set_timeleft 2m
        return "@STD@"
    }
}
```

The argument of `vtk_tasker_set_timeleft` is one of:

- The string "UNLIMITED", meaning that the tasker accepts jobs of any length
- A positive integer
- A time specification
- The number 0, in which case the tasker is effectively suspended, because it does not accept any job

```
...
# -- During the day, suspend the tasker.
vtk_tasker_set_timeleft 0
...
```

Quantization of "timeleft" and "xdur"

For efficiency of the server-tasker messaging interface, all finite values passed to `vtk_tasker_set_timeleft` are silently quantized to the next largest level according to the following table:

Time Left	Quantized value
0-60s	Round up to the next multiple of 5s
60s-30m	Round up to the next multiple of 1m
30m-1h	Round up to the next multiple of 2m
1h-3h	Round up to the next multiple of 5m
3h-6h	Round up to the next multiple of 10m
6h+	Round up to the next multiple of 30m

Troubleshoot Taskers

Troubleshooting TASKERS on UNIX using rsh

If you cannot connect taskers to a server, check your license file with `rlmstat`. The total number of taskers you can connect to VOV servers is limited by the license.

To start a tasker on a remote machine, `vovtaskermgr` uses `rsh` as determined by `vovrsh`. However, a problem with your account setup can prevent remote taskers from starting. The following tests check if your account is setup to run `rsh` correctly:

- ```
% rsh hostname date
```

This test should give you the date without asking for a password and without any other message. If a password is required, adjust the `.rhosts` file in your home directory. If any extra message is displayed, consider making the appropriate changes in your `.cshrc` file to eliminate them.

- ```
% rsh hostname vovarch
```

This test should give you the platform name for the remote host, nothing else. If this message does not appear, verify that your `.cshrc` file is sourcing the `.vovrc` file, even for non-interactive shells. (A common trick in `.cshrc` design is to exit early when it is discovered, by testing for existence of the variable `$prompt`, that the shell is non-interactive. If the exit keyword exists in your script, move the sourcing of `.vovrc` before it.)

- ```
% rsh hostname
```

(A login message appears at this point.)

```
% vovtasker
```

(The usage message from `vovtasker` appears.)

If this test succeeds and the previous test does not, then your `PATH` variable is incorrectly set in `.login`, while it is more appropriate to set it in the `.cshrc` file. You must correct this error before you can use VOV effectively.

- ```
% vovtaskermgr TEST host1
```

This tests if `rsh` works for a list of selected hosts. A host can pass this test and still not qualify as a tasker. This can happen, for instance, if the host does not mount the file system where the project data are stored.

- ```
% vovtaskermgr START -slow
```

This shows in detail how taskers are started. The verbose output may help you debug the problem with starting taskers.

## Troubleshooting TASKERS on UNIX using ssh

Many organizations are now switching to SSH for security reasons. If you also want to use SSH instead of RSH you need to do the following:

```
% vovsshsetup
```

Then in the `taskers.tcl` file, use the option `-rshcmd ssh`:

```
vtk_tasker_set_default -rshcmd "ssh"
```

Now do the tests to see the rest of the setup:

```
% ssh hostname date
% ssh hostname vovarch
```

## vovtsd (Tasker Service Daemon)

The program `vovtsd` is a daemon written as a Tcl script that runs using the VOV `vtclsh` binary. This daemon can be used also to start various types of agents on any type of Windows or UNIX machine.

In Windows, `vovtsd` is started from the command line and then runs in a Windows `cmd` shell. Each `vovserver` connects to `vovtsd` via TCP/IP to start the `vovtasker` process.

## vovtsd

This utility listens for requests to launch taskers for various projects, but always for the same user. The requests typically come from `vovtaskermgr`.

## Usage

```
vovtsd: Usage Message

VOVTSDD: Vov Tasker Service Daemon
 This utility listens for requests to launch taskers for
 various projects, but always for the same user.
 The requests typically come from vovtaskermgr.

USAGE:
 % vovtsd [OPTIONS]

OPTIONS:
 -v -- Increase verbosity.
 -h -- Print this help.
 -debug -- Generate verbose output.
 -help -- This message.
 -normal -- Start a normal daemon (for current user)
 -expire <TIMESPEC> -- Exit from vovtsd after specified time.
 -user <user> -- Specify the user that should be impersonated.
 vovtsd computes the port number by
 hashing the user name.
 -port <n> -- Specify port to listen to.
```

```
-requireauth <n> -- Require key-based auth from clients.
 Client public keys need to be set in
 VOVTSO_USERKEYS env var if enabled.
-userkeys "<key1> <key2> ..." -- Specify a list of public keys that are
 allowed to authenticate the client.
```

EXAMPLES:

```
% vovtsd -normal
% vovtsd -port 16666
% vovtsd -user john -port 16000
```

## Start a Remote vovtasker with vovtsd

To use vovtsd on Windows, follow these steps:

1. Start a command shell on the Windows workstation as the user who is supposed to run the taskers. See below if this user needs to be different from the user logged in on the screen.
2. Set up the shell to use VOV with the vovinit command. This sets the needed environment variables, including PATH.

```
c:\temp> \<install_path>\win64\bat\vovinit
```

3. Mount all filesystems using the appropriate drive letter; these need to agree with the values of serverdir and vovdir in the taskerRes.tcl file for the VOV project.
4. Start vovtsd, possibly using a new window. The -normal option says to use a TCP/IP port calculated from the username. You may specify the port explicitly by using the -port option.

```
c:> start vovtsd -normal
```

## Run vovtsd as a Different User

The vovtasker started by vovtsd will run as the user running vovtsd. If you need for this to be different from the user logged in at the keyboard and screen, you have several options.

On Windows 2000 and Windows XP, you can use the runas.exe command included with the operating system. For example, on Windows XP, logged in as 'user1', you could start a command shell using:

```
C:\temp> runas /user:domain-name\username cmd
```



**Note:** You may need to mount the filesystems for that user. On Windows NT, the drive letters are shared, but on later versions, each user can mount a different filesystem on a given drive letter.

## vovtsd on UNIX

### UNIX: Start a Remote Tasker with vovtsd

The program `vovtsd` can also be used on UNIX. In most cases, you may want to use built-in operating system commands like `rsh` or `ssh`.

You must change the `taskers.tcl` file to set the `-vovtsdport` variable to the port number used by your `vovtsd` daemon. For example, if `vovtsd` is running on port 16001, then your `taskers.tcl` file needs to specify the port number and a number of directories and files **as seen by the remote host**:


```
vtk_tasker_set_default -vovtsdport 16001 -vovdir $env(VOVDIR) -
serverdir
/home/john/vov -logfile /home/john/vov/$env(VOV_PROJECT_DIR).swd/logs/taskers/
@TASKERNAME@/log -executable vovtaskerroot
```

The `vovtaskermgr` command tries to start a remote tasker first with `vovtsd` and then with either `rsh` or `ssh`. Which remote shell command is tried depends on which you have configured with the `-rshcmd` option of `vtk_tasker_define` in the `taskers.tcl` file.

## Map Host Names

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

This is accomplished by overriding the procedure `mapHostName` which in its default implementation simply returns the short host name in all lower case.

 **Note:** A fully qualified hostname is converted to its short name.

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

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

```
mapHostNameInit <context>
```

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

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

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

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

The `mapHostName` procedure is used in Allocator:

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

Monitor includes the procedure `LMnormalizedName`, which is used to eliminate non-ASCII characters from user names, host names, and feature names.

# Resource Management

Altair Accelerator includes a subsystem for managing computing resources. This allows the design team to factor in various constraints regarding hardware and software resources, as well as site policy constraints.

This mechanism is based on the following:

- Resources required by jobs
- Resources offered by taskers
- Resource maps, as described in the file `resources.tcl`

There are several types of resources, which are listed below:

| Resource type                                  | Representation             | Explanation                                                                                                                                                                                                                                                   |
|------------------------------------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Job Resources                                  | <code>name</code>          | A resource required by a job. If the quantity is not shown, the default is 1; "unix" is the same as "unix#1".                                                                                                                                                 |
| Uncountable Resources (also called Attributes) | <code>name</code>          | These resources represent attributes of a tasker that are not countable. For example, a tasker may have attributes such as "unix" or "linux". The quantity is not shown for these resources and it defaults to MAXINT; "unix" is equivalent to "unix#MAXINT". |
| Quantitative Resources                         | <code>name#quantity</code> | Example: The resource <code>RAMTOTAL#2014</code> on a tasker indicates the total amount of RAM on that machine. On a job, it says that the job requires at least the shown amount of RAMTOTAL.                                                                |
| Consumable Resources                           | <code>name/quantity</code> | Example: <code>RAM/500</code> assigned to a job indicates that the job consumes 500 MB of the consumable resources RAM.                                                                                                                                       |
| Negated Resources                              | <code>!name</code>         | Example: "unix !linux" on a job indicates that the job requires a UNIX machine but not a Linux one.                                                                                                                                                           |

The definition of the quantity is related to the context of the resource. If the context is a tasker, quantity represents how much of that resource is available from the tasker. If the context is a job, quantity represents how much of that resource is required by the job.




**Note:** Negated resources are allowed only for the context of a job.

The *unit of measure* is determined by convention for each resource. For example, the resource `RAMTOTAL` is measured in MB. By default, quantity is assumed to be 1; the notation `foo` is equivalent to `foo#1`.

A *resources list* is a space-separated list of resources, which are typical resources offered by the taskers. The following example indicates that a job requires at least 128 MB of RAM and a UNIX host, but not a Linux host.

```
RAMTOTAL#128 unix !linux
```

A *resources expression* is a space separated list of resources and operators: typical resources requested by the jobs or mapped in the resource map set. Operators can be one of the following: `<blank space>`, `&`, `|`, `OR`, `AND`, `!`, and `NOT`. The operators are defined in the table below.

 **Note:** Logical `AND` has precedence over logical `OR` operations.

| Operator                         | Description               |
|----------------------------------|---------------------------|
| <code>&lt;blank space&gt;</code> | implicit logical AND      |
| <code>&amp;</code>               | explicit logical AND      |
| <code>AND</code>                 | explicit logical AND      |
| <code> </code>                   | explicit logical OR       |
| <code>OR</code>                  | explicit logical OR       |
| <code>!</code>                   | explicit logical negation |
| <code>NOT</code>                 | explicit logical negation |

For example, a job may have the following resource requirements:

```
RAMTOTAL#128 unix !linux | RAMTOTAL#512 & linux
```

This job requires either a UNIX host with at least 128 MB of RAM, but not a `linux` host or a Linux host with at least 512MB of RAM.

## Use of Tasker Lists in Resources

The recommended use case is to have one tasker list reference as the first resource in the list. This narrows the scope of the scheduler and improves performance which is the main goal of tasker lists.

Use of tasker lists within resource expressions is not recommended. In particular the use of `OR` operators with tasker lists can result in unexpected behavior.

## Resource Monitoring

Table 2: Summary of Methods to Monitor Resources

|         |                                                                     |
|---------|---------------------------------------------------------------------|
| CLI     | vsm -r<br>vsm -panel resmaps                                        |
| GUI     | In the VOV Monitor dialog, click on the Resources tab               |
| Tcl     | vtk_resources_get (old)<br>vtk_generic_get resourcemaps array (new) |
| Browser | Visit the Resources page                                            |

## Resource Map Set

The Resource Map Set represents the collection of resource maps that have been assigned to a project and the constraints associated with such resources.

Each resource map is described in the file `resources.tcl`, by means of the Tcl procedure `vtk_resourcemap_set`. This procedure requires the following arguments:

1. The name of a resource (in the form "type:name" with optional "type:").
2. The number of units available for the resource, which is either a positive integer or the keyword "unlimited" (case insensitive).
3. The third argument is optional. If it exists, it is used to indicate the name of a resource expression into which the first resource is to be mapped.
4. The fourth argument is also optional; it represents the number of units for that resource that are currently in use. This argument is intended to be used only by [vovresourced](#).
5. The fifth argument is also optional; it represents the expiration date that show when this resource expires (and is automatically removed from the set).

### Example

```
-- We have 4 licenses of DesignCompiler (Synopsys)
vtk_resourcemap_set License:dc_shell -max 4

-- We have 1 license for PathMill (Synopsys), it can only
-- run on Linux machines
vtk_resourcemap_set License:pathmill -max 1 -map linux

-- Specify a resource as the sum of 2 other resources
vtk_resourcemap_set License:msimhdl -sum -map 'License:msimhdlsim OR
License:msimverilog'
```



## Resource Mapping

As the vovservers determines which tasker is most suited to execute a particular job, it performs a *mapping* of the job resources, followed by a *matching* of the mapped resources.

When dispatching a job, the vovservers does the following:

- Gets the list of resources required by a job.
- Appends the resource associated with the priority level, such as `Priority:normal`.
- If it exists, it appends the resource associated with the name of the tool used in the job (reminder: the tool of a command is the tail of the first command argument after the wrappers). The tool resource has type `Tool` and looks like this: `Tool:toolname`.
- Appends the resource associated with the owner of the job, such as `User:john`.
- Appends the resource associated with the group of the job, such as `Group:time_regression`.
- Expands any special resource, i.e. any resource that starts with a "\$".
- For each resource in the list, the vovservers looks for it in the resource maps. If the resource map is found and there is enough of it, that is, the resource is available, the vovservers maps the resource. This step is repeated until one of the following conditions is met:
  - The resource is not available. In this case, the job cannot be dispatched and is left in the job queue.
  - A cycle in the mapping is detected; in this case the job cannot be dispatched at all and is removed from the job queue.
  - The resource is not in the resource map.
- VOV appends the resource associated with the expected job duration to the final resource list. For example, if the job is expected to take 32 seconds, the resource `TIMELEFT#32` will be appended.
- Finally, the vovservers compares the resulting resource list with the resource list of each tasker. If there is a match - all resources in the list are offered by the tasker - the tasker is labeled as eligible. If there is no eligible tasker, the job cannot be dispatched at this time and remains in the queue; otherwise, the server selects the eligible tasker with the greatest effective power.

### Local Resource Maps

Resource maps can be designated as *local*, using the `local` flag.




**Important:** This flag is only available and supported for a FlowTracer installation, utilizing `vovwxd` and an LSF interface.

Resource maps designated as local will be managed on the "local" (FT) side of the `vovwxd` connection instead of the normal case where resource specifications are expected to be managed on the base queue side.

For example, to limit jobs to running 5 at a time from a specific FlowTracer project, do the following:


1. Enable local resources with `run: vovservermgr configure vovwxd.localresources 1`

 **Note:** Alternatively, you can add the following to the `policy.tcl` file:

```
set config(vovwxd.localresources) 1
```

## 2. Create the local resource.

- a. Run `vovresourcemgr set mylocallimit -max 5 -local`

 **Note:** Alternatively, you can add the following to the `resource.tcl` file:

```
vtk_resourcemap_set mylocallimit -total 5 -local
```

This results in limiting running jobs with the local resource `mylocallimit` to a maximum of 5 jobs at a time.

For example, FDL to use a local resource named "mylocallimit":

```
R mylocallimit
J vov /bin/sleep 0
```

## Limitations on the number of Resource Expressions

While the number of OR expressions allowed in a job resource request is limited and controlled by a policy setting, the number of AND expressions including plain expressions without an explicit AND, also has limits.

The scheduler evaluates the resource expressions counting them as it goes. For example a list of 4 separate resource requests (without maps) will result in a max count of 7; 4 from the explicit resource requests and 3 from the automatically added resources (Group, Priority and User). Traversal into a resource map increments the count and a return from a resource map restores the count to value prior to the traversal into the map.

During this traversal, OR operators are recorded and used to influence scheduler operation. The count represents the cumulative depth of the resources. Whenever the count exceeds 30, any traversal that increases the depth is curtailed. This is done silently.

Any OR operators that occur at a depth deeper than 30 are ignored and those scheduling solution are effectively ignored by the scheduler resulting in unexpected behavior.

Large numbers of resource expressions do impact scheduler performance. The general guidance is to keep the explicit resource expressions to fewer than 8 including any mapped ones.

For applications that need a much larger number and where the depth may exceed the 30 limit, it is recommended to place the OR operators early in the resource requests (for example, the left hand side) and to place large numbers of ANDed resource into a resource map.

## Resource Reservation

Resource reservation ensure that specific groups always has access to licenses, regardless of the number of jobs.

For example, if you have 10 licenses of spice but you want to make sure that the 'Production' group has always access to 2 of those licenses, you can use resource reservation.

A resource map can be reserved in whole or in part for a group, a user, or a specific job. When a resource is reserved for a group, jobs in other groups cannot use that resource. Similarly, the same happens for user reservation.

With resource reservation, you are willing to take a hit in the overall utilization of the resources in order to provide instant access to a specific group or user. The reserved resources remain idle even if there is demand for those resources by other groups or users. If you care about overall utilization, consider the other mechanisms of FairShare and preemption.

This functionality is available only to ADMIN users by means of the procedure `vtk_resourcemap_reserve` which accepts the following arguments:

|                               |                                                                                                                                                                                                                                              |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ResourceName</b>           | The name of the resource you want to reserve.                                                                                                                                                                                                |
| <b>TypeOfReservation</b>      | One of the following strings "USER" "GROUP" "JOBID" "JOBCLASS" "JOBPROJ"                                                                                                                                                                     |
| <b>NameOfUserOrGroupOrJob</b> | Depending on the value of the previous argument, this is the name of the user or the name of a group or the ID of the job for which the resource map is to be reserved. The reservation for a specific job is used by the preemption daemon. |
| <b>HowMany</b>                | The quantity of resources to be reserved. If this is greater or equal than the total available quantity, then the whole resource map is reserved.                                                                                            |
| <b>Duration</b>               | The duration of the reservation. After the expiration of the reservation, the resources become available to all.                                                                                                                             |
| <b>Explanation</b>            | A message to document why this reservation has been placed.                                                                                                                                                                                  |

### Examples of `vtk_resourcemap_reserve`

```
Reserve 1 License:abc to user John for the next 3 days.
vtk_resourcemap_reserve License:abc USER john 1 3d "decided by the boss"

Reserve 3 licenses for a group
vtk_resourcemap_reserve License:abc GROUP /app/alpha 3 8h "agreed by the board"

Reserve 3 licenses for a jobclass
vtk_resourcemap_reserve License:abc JOBCLASS hsim 3 8h "as discussed in meeting"
```

## Cancel a Resource Reservation

To cancel all resource reservations, use `vtk_resourcemap_clear_reservations`

```
Cancel all reservations on a resource.
vtk_resourcemap_clear_reservations License:abc
```

A possible way to cancel a specific reservation is to make a new reservation that expires very quickly.

```
Cancel a reservation for user john.
vtk_resourcemap_reserve License:abc USER john 1 2s "cancel"

Cancel a reservation for a group
vtk_resourcemap_reserve License:abc GROUP /app/alpha 3 2s "cancel"

Cancel a reservation for a jobclass
vtk_resourcemap_reserve License:abc JOBCLASS hsim 3 3s "cancel"
```

## Delete Resources

To delete a resource map, use the Tcl procedure `vtk_resourcemap_delete` in `vovsh`. Resource maps may be deleted by ID or by name.

For example:

```
Delete a resource by name
vtk_resourcemap_delete License:dc_shell

Delete a resource by id
vtk_resourcemap_forget 00023456
```

Resource maps are automatically deleted after they expire, as soon as no job is using the resource.

The `License:` resources that are maintained by `vovresourced` and by `Allocator` are created with relatively short (typ. 5m) expiration times, but these are intended to be several times the duration at which they are refreshed by the daemon from license in-use information.

Rarely, you may observe jobs where `nc info` or the browser UI will show waiting on INFO: being deleted. This is shown when a job is waiting on a resource map that is in the process of being deleted or expiring.

## Rank the Resource Agents

Resource ranking allows a resource to be managed by multiple agents, such is the case with `vovmultiqueued` and `vovresourced`. The simple idea is that a ranking is assigned to the agents, and the agent with the highest rank wins. With ranking, you can setup a redundant system of daemons.

By default, `vovmultiqueued` has rank 20 while `vovresourced` has rank 3. If a resource is managed by both, then `vovmultiqueued` wins. However, if the `vovmultiqueued` daemon stops (for maintenance or

because the network link has been broken) then `vovresourced` will take over after the expiration time of the resource.

Typically, agents with higher ranking will use shorter expiration times.

### Control the Rank of `vovresourced`

The default rank for `vovresourced` is 3. From the command line, use the option `-rank` as in:

```
% vovresourced -rank 4
```

or set the variable `RESD(rank)` in the `config.tcl` file.

### Control the Rank of `vovmultiqueued`

The default rank for `vovmultiqueued` is 20. From the command line, use the option `-rank` as in:

```
% vovmultiqueued -rank 28
```

or call the procedure `VovMQSetRank` in the `config.tcl` file.

```
Fragment of vovmultiqueued/config.tcl
VovMQSetRank 28
```

### Ranks Less Than 20 and Out-of-queue Jobs

If an agent has a rank less than 20, then the server will automatically adjust the number of available resources to account for out-of-queue jobs. This also means that for rank 20 or above, the server does not automatically account for out-of-queue jobs.

This is done because in the case of `vovmultiqueued`, the server does not have enough information to compute the out-of-queue jobs, and such computation is therefore left to `vovmultiqueued`.

### Rank 1000 and More

This rank is used for resources in the dedicated server for Allocator.

### Summary of Ranks for Resource Maps

|              |                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------|
| 0            | Used for automatically generated resources                                                       |
| 1-19         | Used for <code>vovresourced</code> . Default is 3                                                |
| 20-999       | Used for resources managed by Allocator. Default is 30 and typical values are between 20 and 40. |
| 1000         | Used for resources in the dedicated server for Allocator.                                        |
| other values | Not used                                                                                         |

# Resource Daemon Configuration

## vovresourced

Table 3:

|                       |                                    |
|-----------------------|------------------------------------|
| Working directory     | vnc.swd/vovresourced               |
| Config file           | vnc.swd/vovresourced/resources.tcl |
| Auxiliary config file | \$VOVDIR/local/resources.tcl       |
| Info file             | vnc.swd/vovresourced/resourced.pid |

The daemon `vovresourced` is the main agent that defines the resources of the vovservers. The configuration file is `resources.tcl`, which is located in the server configuration directory. This file defines which resources are to be used by the server by calling the procedure `vtk_resourcemap_set`. Examples are available in the `vnc.swd/resources.tcl` file.

The procedures `vtk_flexlm_monitor` and `vtk_flexlm_monitor_all` are used to define resources that are derived from licenses. If the `resources.tcl` file calls for monitoring FlexNet Publisher features with the command `vtk_flexlm_monitor`, or when Monitor receives notification of an event from LICMON, `vovresourced` then retrieves the information about the event. Refer to the `vnc.swd/resources.tcl` file for examples.

## Refresh Rate

The frequency of the checks can be configured in the `resources.tcl` file as shown below:

```
set RESD(refresh) 10000; #Set refresh time to 10,000 milliseconds.
```

## Starting vovresourced

The program `vovresourced` is normally started automatically by the server. `vovresourced` can also be invoked manually from any directory. After reading the `vovresourced` configuration file, `vovresourced` then reads the auxiliary configuration file `$VOVDIR/local/resources.tcl` (if the auxiliary file has been created). When the `resources.tcl` file is changed, the `vovresourced` daemon is restarted with the following commands:

```
% vovproject reread ; # Generic method for any vovserver.
% nc cmd vovproject reread ; # Specific for Accelerator.
% ncmgr reset ; # Specific for Accelerator.
```

## vovresourcemgr

vovresourcemgr is a utility for managing VOV resource maps. It may be used to create, modify, forget, and reserve resource maps.

The vovresourcemgr utility command connects to and acts on the VOV project enabled in the shell where it is launched. To act on Accelerator, use vovproject enable vnc, or precede it with nc cmd as shown in the examples below.

```
vovresourcemgr: Usage Message
```

```
USAGE:
```

```
% vovresourcemgr COMMAND [options]
```

```
COMMAND is one of:
```

```
show Show summary info about all resource maps
show [R1..RN] Show info about specified resource map(s)
matches RESMAP Show license matching info
ooq RESMAP Show out of queue license handles
create RESMAP map-options
 Create a new resource map
set RESMAP map-options
 Create a new or modify an existing resource map
reserve RESMAP TYPE WHO HOWMANY HOWLONG WHY [-exclusive]
 Place a reservation on a resource map
forget [-force] R1 [R2..RN]
 Remove resource map(s) from the system
```

```
MAP-OPTIONS:
```

```
-expire specify expiration (timespec) relative to now
-max specify quantity
-map specify map-to value
-rank specify rank when setting
-noooq do not track out-of-queue
-local specify that this is a local resource (when using vovwxd)
```

```
For reserve, TYPE is one of: USER, GROUP, JOBCLASS, JOBPROJ, JOBID.
```


```
EXAMPLES:
```

```
% vovresourcemgr show
% vovresourcemgr show Limit:abc
% vovresourcemgr matches Limit:abc
% vovresourcemgr create License:spice -max 8
% vovresourcemgr set License:spice -max 10
% vovresourcemgr set License:spice -map "Policy:spice"
% vovresourcemgr ooq License:spice
% vovresourcemgr reserve License:spice USER john,jane 3 3d ""
% vovresourcemgr reserve License:spice USER bill 1 1w "" -exclusive
% vovresourcemgr forget License:spice
% vovresourcemgr forget -force License:spice
```

```
% vovresourcemgr show ; # show defined resource maps
% vovresourcemgr show R ; # show details of resource map R
% vovresourcemgr set ; # set/create a new resource map
% vovresourcemgr reserve ; # reserve resource map for user or group
% vovresourcemgr ooq ; # show Out-Of-Queue ooq (for Accelerator)
% vovresourcemgr forget ; # remove a resource map from vovserver
```


## Dynamic Resource Map Configuration

Persistent resource maps are defined in the `resources.tcl` configuration file for a project. The `vovresourcemgr` command is useful to make changes to the resource maps on the fly.

 **Note:** Unlike resource maps defined in `resources.tcl`, changes made with `vovresourcemgr` do not persist across restarts of `vovserver`.

The `create` command checks for existence of the named resource map and exits with a message if it already exists. The `set` command will create or replace an existing resource map with the given values with no confirmation.

The following example creates a new resource map named `Limit:spice`, which is created with a quantity of 10 and an empty map-to value.

 **Note:** `nc cmd` shows this example is to act on Altair Accelerator.

```
% nc cmd vovresourcemgr set Limit:spice 10 ""
```

## Resource Map Reservation

Following is an example of using `vovresourcemgr` to place a reservation on a resource map. In this case, two of the resource maps called `License:spice` are reserved for user `john` for an interval of 4 hours. The resource map reservation will automatically expire after 4 hours.

```
% nc cmd vovresourcemgr reserve License:spice USER john 2 4h "library char"
```

## Workaround for Misspelled Resource

Sometimes users submit jobs to Altair Accelerator that request nonexistent resources, which causes the jobs to be queued indefinitely. Such jobs can be made to run by creating the missing resource, or by modifying the jobs to request the correct resources.

The following example creates four temporary `License:sspice` resources that are mapped to the correct `License:spice` resource. `License:sspice` is an incorrect request - that resource does not exist. A temporary resource is created with that name that will be mapped to the correct resource, `License:spice`

```
% nc cmd vovresourcemgr create License:sspice -max 4 "License:spice"
```



## Forget Unneeded Resource Maps

Continuing the above example - the temporary resource map may be removed after the malformed jobs have run. Or, you can just let it expire.



**Note:** There is no confirmation; the command acts immediately.

```
% nc cmd vovresourcemgr forget License:sspice
```

## Policies and Resources

You can define site-specific policies on how resources are allocated to various projects by creating a file called `$VOVDIR/local/resources.tcl`. This file contains the definition of a procedure called `vtk_resources_policy_hook` and can be used to limit the amount of resources assigned to a given project.

The default definition for this procedure is:

```
Default definition
proc vtk_resource_policy_hook { projectName resName max } {
 return $max
}
```

For example, if you have 70 license of SPICE and do not want any project to have more than 20, except special projects, you can write the following definition for `vtk_resources_policy_hook`:

```
Example of definition of vtk_resource_policy_hook
proc vtk_resource_policy_hook { projectName resName max } {
 switch -glob $resName {
 "spice*" {
 switch -glob $projectName {
 "special*" { return $max }
 default {
 return [expr $max > 20 ? 20 : $max]
 }
 }
 }
 default {
 return $max
 }
 }
}
```

## Add Resources

Generic resources are added to Altair Accelerator via the `vtk_resourcemap_set` procedure call:

```
vtk_resourcemap_set <name> <quantity> [map]
```

Below are two examples:

```
vtk_resourcemap_set myres 2
vtk_resourcemap_set myunlimitedres UNLIMITED
```

### Example: Node Locked License

In the scenario of this example, a license does not utilize FlexNet Publisher or another dynamic license management solution, but does require a tool to run only on one specific host. In this example, the tool is `spice`, the host is `pluto`, and the license is for two concurrent instances of `spice`. Following are the steps to correctly handle this constraint:

1. Choose a name (in the form `name` or `type:name`) to represent the node locked resource (such as `License:spice_pluto`) and a name to be announced to the users (such as `License:spice`). In this way, it is hidden to the users that the `spice` license is locked to a given node. Also, if there is an upgrade to a floating license or multiple node-locked licenses, that can be carried out without having to announce it to the users.
2. Add the following lines to the `resources.tcl` file:

```
vtk_resourcemap_set License:spice UNLIMITED License:spice_pluto
vtk_resourcemap_set License:spice_pluto 2 pluto
```

```
% nc cmd vovproject reread
```

3. Let the job declare that it requires the resource `License:spice`; use option `-r` in `nc run` as shown below:

```
% nc run -r License:spice -- spice -i chip.spi
```

## CHOICEMASK Resource

The resource CHOICEMASK can be found, typically, in resource expressions assigned to "resumer jobs", which are auxiliary jobs used to resume other jobs that have been preempted. This is not really a resource, but rather a method to control the evaluation of the expressions that contain OR branches.

Normally, all branches of the expression are evaluated. If CHOICEMASK is present in the expression, it controls which branches are chosen. For example, if you have an expression with one OR branch, the presence of CHOICEMASK#0 forces the selection of the left branch, while CHOICEMASK#1 forces the selection of the right branch.

If more OR branches are present in the expanded expression, the bits in CHOICEMASK are used in order to select either the left (bit value 0) or the right (bit value 1) branch in the expression.

It is unlikely that a user will want to use CHOICEMASK to a job, but it is considered to be legal.

## Examples of CHOICEMASK

The expression

```
"License:aa OR License:bb"
```

evaluates first the availability of License:aa, then the availability of License:bb.

The expression

```
"CHOICEMASK#0 (License:aa OR License:bb)"
```

forces the choice of License:aa.

The expression

```
"CHOICEMASK#1 (License:aa OR License:bb)"
```

forces the choice of License:bb.

## License-based Resources

This section describes the Accelerator interface to Monitor, an application that monitors license servers and makes the in-use information available to Accelerator.

Many software products in Electronic Design Automation use FlexNet Publisher licensing by Flexera or other vendor-specific license mechanisms.

Monitor provides a centralized interface between vendors' license daemons and Accelerator. The benefits of this approach are:

- Faster response, with reduced load on the license daemons
- Improved consistency of license in-use information
- Individual projects need not be concerned with license details
- Browser-based interface to access information about licenses in-use

Accelerator is shipped with an edition of Monitor that is licensed to monitor **current license activity only**, and provide that information to Accelerator. This edition is referred to as LMS (Monitor Small).

Monitor can also store, report, and graph historical license usage and denial activity.



**Note:** A full Monitor license from Altair is required to enable historical and denial information.

## Configuration

Refer to *Installation Guide* for details about installing and configuring the Monitor product.

By default, Accelerator assumes that the Monitor server is running on the same machine as the Accelerator server on port 5555, under the VOV project name of `licmon`. If this is not the case, the following statements will need to be placed at the top of the `resources.tcl` file to inform the Accelerator resource daemon of Monitor's details:

```
Enable Accelerator to see Monitor.
This is a security feature.
set lm(ssl) true

Fragment of resources.tcl
This is the default configuration.
set LM(flexlmd) localhost:5555
set LM(licmon) licmon

You may want to specify a different Monitor
running on a different host, a different port
with a different name.
set LM(flexlmd) someOtherHost:25555
set LM(licmon) licmonTest
```

Accelerator uses `vtk_flexlm_monitor` and `vtk_flexlm_monitor_all` statements in its `resources.tcl` configuration file in order to communicate with the Monitor product to obtain license utilization information.

The procedure `vtk_flexlm_monitor` takes from one to three arguments:

1. **feature** - the name of the license feature. This is the name of any feature monitored by Monitor. The name may include a specific Monitor tag. If no tag is specified, the cumulative count for all tags containing the feature will be used.
2. **resource** - the Accelerator resource name. If the resource name is specified, then this name is the actual resource name used. If the resource name is not specified, the name defaults to `License:<feature>`.
3. **map** - an optional resource to which the resource should be mapped.

For example:

```
vtk_flexlm_monitor Design-Compiler

Pick up a specific tag for Design-Compiler
This maps the feature Design-Compiler to License:Design-Compiler
vtk_flexlm_monitor SNPS/Design-Compiler

Specify a different resource name
vtk_flexlm_monitor SNPS/Design-Compiler License:dc

Additionally specify that all jobs using License:dc need to also use
a linux resource.
vtk_flexlm_monitor SNPS/Design-Compiler License:dc linux
```

As an alternative to individually calling `vtk_flexlm_monitor` for each feature to monitor, `vtk_flexlm_monitor_all` can be used. The default behavior of this procedure is to create resources for all features that are known to Monitor. This procedure has a number of options:

Table 4: *vtk\_flexlm\_monitor\_all* Options

| Option                          | Description                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-daemon host:port</code>  | Specifies the host and TCP/IP port where Monitor is to be contacted to get license data via HTTP. If not given, the procedure reads the <code>info.tcl</code> in <code>\$VOVDIR/../../licmon/licmon.swd/vovlmd</code> to locate the daemon. If the daemon cannot be located, the procedure returns 0.                                                                                        |
| <code>-tag tag</code>           | Only use features from the source having this tag. The default is to use features from all license sources. This option may be repeated.                                                                                                                                                                                                                                                     |
| <code>-tags list_of_tags</code> | Same as above, only with a list of tags to be included. This option may be repeated.                                                                                                                                                                                                                                                                                                         |
| <code>-I regexp</code>          | Appends <code>regexp</code> to the list of regexps evaluated for feature inclusion. If no <code>-I</code> options are given, all features from the given source are included.                                                                                                                                                                                                                |
| <code>-X regexp</code>          | Appends <code>regexp</code> to the list of regexps evaluated for feature exclusion. If no <code>-X</code> options are given, no features are excluded.                                                                                                                                                                                                                                       |
| <code>-It regexp</code>         | Appends <code>regexp</code> to the list of regexps evaluated for tag inclusion. If no <code>-It</code> options are given, all tags from the given source are included.                                                                                                                                                                                                                       |
| <code>-Xt regexp</code>         | Appends <code>regexp</code> to the list of regexps evaluated for tag exclusion. If no <code>-Xt</code> options are given, no tags are excluded.                                                                                                                                                                                                                                              |
| <code>-fproc fproc-name</code>  | Specifies the name of the user-defined Tcl filter procedure to apply to the features. The default is <code>vtk_flexlm_monitor_filter {tag feature}</code> . This procedure takes two parameters, a tag and a feature name, and returns a Boolean, where 1 means to include the feature, and 0 means to exclude it. The default procedure always returns 1.                                   |
| <code>-rproc rproc-name</code>  | Specifies the name of the user-defined Tcl procedure that returns the resource map name for a feature. The default is <code>vtk_flexlm_monitor_resname {tag feature}</code> . This procedure takes two parameters, a tag and a feature name, and returns a string, which is the VOV resource name for the feature. The default procedure prepends <code>License:</code> to the feature name. |
| <code>-mproc mproc-name</code>  | Specifies the name of the user-defined Tcl procedure that returns the right-hand-side of the resource map name for a feature. The default is <code>vtk_flexlm_monitor_mapname {tag feature}</code> . This procedure takes two parameters, a tag and a                                                                                                                                        |

| Option                           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  | feature name, and returns a string, which is the right-hand-side of the VOV resource map for the feature. The default procedure returns "", which means no mapping.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>-order list_of_tags</code> | <p>This options controls the order in which multiple tags (think of "license files") are listed in Accelerator. This applies when there are multiple tags for the same feature. For example, if the feature <code>abc</code> is in two tags, <code>SNPS_BLR/abc</code> and <code>SNPS_US/abc</code>, you will get a resource map called <code>License:abc</code>, which is the OR of the two resource maps associated to each feature, as in</p> <pre>License:abc → License:SNPS_US_abc OR License:SNPS_BLR_abc</pre> <p>However, it is also possible to get the following map, with inverted order of the tags:</p> <pre>License:abc → License:SNPS_BRL_abc OR License:SNPS_US_abc</pre> <p>The <code>-order</code> option allow controlling which map will be used. For example: <code>-order "SNPS_BLR SNPS_US"</code> will use <code>SNPS_BLR</code> before <code>SNPS_US</code>. The option can be repeated multiple times.</p> |

This procedure can make getting started with license management much easier and faster. Use this procedure with caution, especially if it is used with any user-defined Tcl procedures. Place the procedure definitions in the `resources.tcl` file, and be sure to specify the names carefully.

### More examples for `vtk_flexlm_monitor` and `vtk_flexlm_monitor_all`

Fragment of `resources.tcl` file:

```
Monitor the feature PrimeTime
vtk_flexlm_monitor PrimeTime

Monitor the feature PrimeTime, control the order of the tags
vtk_flexlm_monitor -order "SNPS_US SNPS_CH SNPS_FR" PrimeTime

Monitor the feature for Design-Compiler. Internally VOV
uses the token dc_shell_license.
vtk_flexlm_monitor Design-Compiler dc_shell_license

The FlexNet Publisher feature "pathmill" maps to the VOV resource "pathmill"
which, in turn, maps to the resource "sun7"
vtk_flexlm_monitor pathmill pathmill sun7
```

Example : Monitoring all features

```
#
Monitor all the features gathered by the Monitor at grove:5555#
vtk_flexlm_monitor_all -daemon grove:5555
```

```

Monitor all the features gathered by the Monitor,
control the order of the tags

vtk_flexlm_monitor_all -order "SNPS_US SNPS_FR" -order "MGC_US MGC_FR MGC_UK"
```

Example: Monitor some features, user-defined map proc

```

Monitor some features gathered by the LicenseMonitor at grove:5555
Make all the Fintronic tools go to the finfarm vovtaskers
by defining a RHS-procedure

proc fintronic_mapname {tag feature} {
 set rval ""
 if { [regexp {^fin} $feature] } {
 set rval "finfarm"
 }
 return $rval
}

Only monitor features from tags REAL and Altair Accelerator products
vtk_flexlm_monitor_all -daemon grove:5005 -tag REAL -tag RTDA -rproc
 fintronic_mapname
```

In the above example, features related to the Fintronic Finsim Verilog simulator, recognized by the feature name beginning with 'fin', will have a right-hand-side of 'finfarm' added to the resource map, so that such jobs will go to vovtaskers offering the 'finfarm' resource.

## Configuring Job and License Checkout Matching

When a resource is derived from a license feature, it is useful to attempt a matching of the license handles that are currently checked out and the jobs that are currently running.

This is a challenging task because the information relative to the license handles is often incomplete and incorrect. For example, FlexNet Publisher does not report the checkout time to the second, and does not reveal the PID of the process that has requested the checkout. The PID alone would enable a precise matching. Instead, we have to accept the best possible solution based on approximate input data.

The matching is automatically enabled for all licenses.

When the number of running jobs for a given license exceeds about 1,000, the matching becomes onerous on the vovserver, which can be detected by "Long Service" messages in the vovserver log. For this issue, it is recommended to disable the handle matching for selected licenses.



**Note:** If the `rds.enable` configuration parameter is set to 1, then RDS resource management is active. Otherwise, classic resource management is active.

*When RDS Resource Management is Enabled*

Edit the SWD/resources.cfg AVS resource management configuration file in one of two ways to assert the NOMATCH flag.

The recommended way is to assert the NOMATCH flag attribute directly in the feature rule object within the RESOURCE\_MAPS array attribute:

```
{
 LICENSE_MONITORS=[{ NAME="lm_cool_project" }]
 RESOURCE_MAPS = [
 { TAGS="Blue", FEATURES="great_feature", NOMATCH}
]
}
```

Alternatively, the NOMATCH flag can be configured for a license feature using the top-level FLAGS array attribute:

```
{
 LICENSE_MONITORS=[{ NAME="lm_cool_project" }]
 FLAGS = [
 { TYPENAME="License:great_feature", NOMATCH },
]
 RESOURCE_MAPS = [
 { TAGS="Blue", FEATURES="great_feature"}
]
}
```

#### *When Classic Resource Management is Enabled*

Edit the Tcl configuration file SWD/resources.tcl to disable matching for a feature:

```
vovresSetFlags License:great_feature -nomatch

A less preferred method is as follows:
vtk_resourcemap_set License:great_feature -nomatch

The great_feature license feature is created by a vtk_flexlm_monitor
or vtk_flexlm_monitor_all directive.

vtk_flexlm_monitor_all
```

### Configuration Parameters that Control Matching

The following server configuration parameters control, configure, and limit Accelerator's job and license checkout matching algorithm.

|                                               |                                                                                                  |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------|
| <b><i>resusermatchtolerance</i></b>           | In seconds, determines a tolerance in matching checkout timestamps with jobs starts              |
| <b><i>resusermaxmatches</i></b>               | The number of "also" matches that we look for. .                                                 |
| <b><i>resuserDisableMatchingThreshold</i></b> | A threshold for disabling matching if the sum of Monitor handles and FlowTracer jobs exceeds it. |



# IT Administrator Topics

## SSH Setup

If you use SSH to login onto certain machines in your LAN that you want to use as taskers, you need to setup SSH so that it will not ask for your password.

You can do that using the utility `vovsshsetup`:

```
% vovsshsetup
```

Test the setup by running simple commands on the remote machine:

```
% ssh <remotemachine> date
```

### Disable Host Checking for Automatic Tasker Startup

If you use SSH to start many taskers, you may find that for each new tasker you want to start you get this question:

```
The authenticity of host 'hs123 (10.11.12.123)' can't be established.
RSA key fingerprint is dd:af:39:92:c0:72:83:6b:48:25:96:4b:0e:e4:8c:76.
Are you sure you want to continue connecting (yes/no)?
```

To avoid that message and having to type 'yes' for each new host you want to connect, you can do the following:

- Edit the file `~/.ssh/config`
- Add the statement `StrictHostKeyChecking no` to the file

## Network Conditioning

Each computer has its own clock. Clocks on different computers in the same network might not be synchronized. The system administrator should keep all clocks in the network synchronized.

### Synchronizing Networks

The recommended method to keep the clocks synchronized is with the Network Time Protocol (NTP), available on many modern OS's, which can keep the clocks synchronized to within a few milliseconds.

Alternatively, refer to the following table to find the proper synchronization command depending on the operating system:

| OS      | Tool     | Example                                          |
|---------|----------|--------------------------------------------------|
| Linux   | rdate    | <code>rdate -s &lt;host&gt;</code>               |
| Windows | net time | <code>net time \\&lt;&lt;host&gt; /set /y</code> |

| OS | Tool    | Example                                                                                             |
|----|---------|-----------------------------------------------------------------------------------------------------|
|    | Tardis™ | See <a href="https://www.mingham-smith.com/tardis.htm">https://www.mingham-smith.com/tardis.htm</a> |

Whenever a vovtasker connects to the vovserver, the delta between the clocks of the two hosts is computed. Thereafter, the procedure is repeated about every 10 minutes, in order to catch clock drifts that are common in almost all computers.

## File System Offsets

The file system that contains the design files may reside on a machine that is not the same as the one where the vovserver is running. The clocks in the two machines may be different, resulting in an offset. These offsets can be easily avoided. Here we show what can be done to work with an unsynchronized network.

### Measuring the Offset

The complete list of known offsets is available in the Filesystems page.

To check the offset of a particular file system:

1. Login onto the machine that is running the vovserver
2. Connect to a running project
3. Change to a writable directory in the filesystem you want to test
4. Use the command `vovguru` as follows:

```
% vovguru -T .
vovguru message [11:16:49]: Filesystem /name/of/filesystem seems OK (deltaT=0)
```

The program `vovguru` writes a file in the current directory and computes *deltaT* as the difference between the file timestamp and the machine clock.

In this example, there is no offset. Otherwise, you would get a warning and *deltaT* will have a value different from 0. Acceptable offsets may be in the range of +/- 4 seconds. Larger offsets indicate a poorly managed network, and the problem should be corrected by your system administrator, rather than trying to work around it.

### Dealing with the Offset

You have the following options:

- The best option is to synchronize the network.
- The second option is to allow a **tolerance** in the comparison of timestamps, which is accomplished by modifying the `policy.tcl` file.
- The least attractive option, if the offset is **acceptably small**, consists of adding appropriate delays in the job firing and execution procedures. Notice that the delays affect every job, and the entire retracing system becomes slower than it could be if the clocks had been synchronized.

### **The file system clock is running ahead of the server**

If the filesystem clock is ahead of the vovserver, as indicated by a *positive deltaT* value, sequences of jobs are likely to fail because the timestamps of intermediate input files seem more recent than the start date of the job. The failed jobs, if fired again, will typically succeed, because the files have had time to age. To avoid this kind of failures, it is sufficient to delay the firing of the jobs by setting the environment variable `VOV_DELAY_FIRE`. This variable, whose default value is 0, is used by vovtasker to delay job firing.

For example, if the time offset is 2 seconds, you can add the following line to the `setup.tcl` file:

```
setenv VOV_DELAY_FIRE 2
```

and then you must restart the taskers.

### **The file system clock is running behind the server**

If the filesystem clock is running late with respect to the clock in the vovserver, as indicated by a *negative deltaT* value, it is likely that short jobs will fail because the timestamp of some outputs is younger than the job start time, giving the impression that the job has not had any effect on those outputs. The solution is to introduce a delay between the official start of the job, which is the time the job is added to the design trace, and the actual processing by the job. This is accomplished with the environment variable `VOV_DELAY_BEGIN`, whose default value is 0.

For example, if the time offset is 1 second, you can add the following line to the `setup.tcl` file:

```
setenv VOV_DELAY_BEGIN 1
```

Then you need to restart the taskers and source the `setup.tcl` file again in your current shell.

## **NFS and VOV**

Before you can use NFS, be it as server or client, you must make sure your kernel has NFS support compiled in.

### **NFS on Linux Only**

Newer kernels have a simple interface on the proc filesystem for this, the `/proc/filesystems` file, which you can display using `cat`. If `nfs` is missing from this list, you have to compile your own kernel with NFS enabled, or perhaps you will need to load the kernel module if your NFS support was compiled as a module.

## NFS Mounting Options

There are a number of additional options that you can specify to mount upon mounting an NFS volume. These may be given either following the `-o` switch on the command line or in the options field of the `/etc/fstab` entry for the volume. In both cases, multiple options are separated by commas and must not contain any whitespace characters. Options specified on the command line always override those given in the `fstab` file. The following is a partial list of options you would probably want to use:

| Option                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rsize=n</code> and <code>wsiz=n</code> | These specify the datagram size used by the NFS clients on read and write requests, respectively. The default depends on the version of kernel, but is normally 1,024 bytes.                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>timeo=n</code>                         | This sets the time (in tenths of a second) the NFS client will wait for a request to complete. The default value is 7 (0.7 seconds). What happens after a timeout depends on whether you use the <code>hard</code> or <code>soft</code> option.                                                                                                                                                                                                                                                                                                                                                                |
| <code>retrans=n</code>                       | The number of minor timeouts and retransmissions that must occur before a major timeout occurs. The default is 3 timeouts. When a major timeout occurs, the file operation is either aborted or a "server not responding" message is printed on the console.                                                                                                                                                                                                                                                                                                                                                   |
| <code>hard</code>                            | Explicitly mark this volume as hard-mounted. This is on by default. This option causes the server to report a message to the console when a major timeout occurs and continues trying indefinitely (The client will continue to attempt the NFS file operation indefinitely if the operation fails). Hard mounts should be used when the server and the link to the server are known to be reliable. Hard mounts with the interruptible option enabled is the recommended method of mounting remote filesystems.                                                                                               |
| <code>soft</code>                            | Soft-mount (as opposed to hard-mount) the driver. This option causes an I/O error to be reported to the process attempting a file operation when a major timeout occurs ( <code>timeo</code> option). Retries NFS file operation <code>n</code> times ( <code>retrans</code> option) as set by the number of retries before reporting error option; returns error if no server response in <code>n</code> tries. Soft mounts are recommended for filesystems whose servers are considered unreliable, or if the link is slow. Unlike spongy mounts, soft mounts may time out during read and write operations. |
| <code>spongy</code>                          | Sets soft semantics for <code>stat</code> , <code>lookup</code> , <code>fsstat</code> , <code>readlink</code> , and <code>readdir</code> NFS operations and hard semantics for all other NFS operations on the filesystem. Spongy mounts are preferable to soft mounts because spongy mounts will not time out during read and write operations. They are recommended for slow, long-distance, or unreliable links, and for unreliable servers.                                                                                                                                                                |

| Option            | Description                                                                                  |
|-------------------|----------------------------------------------------------------------------------------------|
| <code>intr</code> | Allow signals to interrupt an NFS call. Useful for aborting when the server doesn't respond. |

Except for `rsize` and `wsize`, all of these options apply to the client's behavior if the server should become temporarily inaccessible. They work together in the following way: Whenever the client sends a request to the NFS server, it expects the operation to have finished after a given interval (specified in the timeout `timeo` option). If no confirmation is received within this time, a so-called minor timeout occurs, and the operation is retried with the timeout interval doubled. After reaching a maximum timeout of 60 seconds, a major timeout occurs.

By default, a major timeout causes the client to print a message to the console and start all over again, this time with an initial timeout interval twice that of the previous cascade. Potentially, this may go on forever. Volumes that stubbornly retry an operation until the server becomes available again are called hard-mounted. The opposite variety, called soft-mounted, generate an I/O error for the calling process whenever a major timeout occurs. Because of the write-behind introduced by the buffer cache, this error condition is not propagated to the process itself before it calls the write function the next time, so a program can never be sure that a write operation to a soft-mounted volume has succeeded at all.

Whether you hard- or soft-mount a volume depends partly on taste but also on the type of information you want to access from a volume. For example, if you mount your X programs by NFS, you certainly would not want your X session to go berserk just because someone brought the network to a grinding halt by firing up seven copies of Doom at the same time or by pulling the Ethernet plug for a moment. By hard-mounting the directory containing these programs, you make sure that your computer waits until it is able to re-establish contact with your NFS server. On the other hand, non-critical data such as NFS-mounted news partitions or FTP archives may also be soft-mounted, so if the remote machine is temporarily unreachable or down, it doesn't hang your session. If your network connection to the server is flaky or goes through a loaded router, you may either increase the initial timeout using the `timeo` option or hard-mount the volumes. NFS volumes are hard-mounted by default.

Hard mounts present a problem because, by default, the file operations are not interruptible. Thus, if a process attempts, for example, a write to a remote server and that server is unreachable, the user's application hangs and the user can't do anything to abort the operation. If you use the `intr` option in conjunction with a hard mount, any signals received by the process interrupt the NFS call so that users can still abort hanging file accesses and resume work (although without saving the file).

Allows the user to kill a process that is hung while waiting for a response on a hard-mounted filesystem. This option is useful if the server or the connection to the server is known to be slow or unreliable. It is recommended to always have the `intr` option on. A keyboard interrupt is configured by entering `stty intr key` where `key` is the keyboard key you wish to use to issue an interrupt.

NFS that are mounted `rw` should use the `hard` option (plus eventually the `intr` one).

Do not specify either `soft` and `intr` options when mounting NFS filesystems that are writable or that contains executable files.

## Network Overhead

VOV lets you distribute computation across the network without adding significant overhead. In some cases, it pays to be careful about the distribution of data and computation.

The following table shows the time it takes to archive the same 6.3MB Linux library using the same Linux CPU (called "reno") and varying the location of the modules and that of the target library.

| CPU  | Modules | Library | Time  |
|------|---------|---------|-------|
| reno | tahoe   | tahoe   | 1m42s |
| reno | tahoe   | reno    | 47s   |
| reno | reno    | reno    | 5s    |

In all cases the network is a rather quiet standard 10BaseT Ethernet, the machine "tahoe" is a Sparc5/110 running Solaris2.5 (that is a sun5), and the command to build the library was of the type:

```
% ar rscv LIBRARYNAME.a *.o
```

Building the library across the network takes almost 2 minutes, while doing the same operation locally takes just 5 seconds. The difference is explained by extra disk and network traffic. Just copying the object files from the sun5 to the Linux box takes 29 seconds.

For other operations, such as linking of large binary files, the difference may be even more dramatic. The point is that, if you want speed, you must limit the network traffic. With VOV this is easy and can be made completely transparent to the end users.

You allocate disk space on various machines in the network and, as part of your design methodology, you include operations that move the data intelligently between the disks. It is easy to constrain some operations to be performed on the machine that physically owns the data; use the Job Editor and modify the Resources field. For example, if the Linux box with the disk space is called "reno," you can restrict selected tools to run on it by setting the Resources field to reno.

## Using PAM (Pluggable Authentication Modules) for Browser Authentication

### Login using Web Browsers

For CLI and GUI commands, the user is already authenticated by the operating system login, and VOV programs are run as the logged-in user.

The `vovserver` program natively supports HTTP for VOV's browser-based user interface. When a browser connects to the `vovserver`, the user may wish to authenticate as one other than the user running the browser program on the remote host. Also, remote users can have complete administrative control of the host where the browser runs, and be spoofing another user's login name. Hence, VOV

asks for a username and password, and uses these to authenticate the user on the host where the `vovserver` runs.

Once a user has authenticated with a given login name, their VOV privilege level with respect to VOV operations is determined by the `security.tcl` configuration file for that `vovserver`. Please refer to [VOV Security](#).

## Authentication Mechanisms

The `vovserver` is a PAM-aware application: its authentication may be changed without recompiling the program. On Linux and MacOS-X, the `vovserver` uses `vovpamauth` which in turn uses PAM to authenticate users.

On other platforms, `vovserver` uses the regular `crypt()` method to accept the provided password.

## Authentication Control

To enable or disable PAM, you can do so with the variable `config(enablePam)` in the `policy.tcl` file.

```
force PAM to be used when available
set config(enablePam) 1
```

Alternatively, you can control the use of PAM with `vovsh` and `vtk_server_config`. You must select which `vovserver` to act on by using the `vovproject enable` command first.

```
% vovproject enable some-ft-project
% vovsh -x 'puts [vtk_server_config enablepam 1]'
```



**Note:** When PAM is enabled, the `vovpamauth` program scans, at startup, the directory `/etc/pam.d`, searching in order for files named 'vovauth', 'ftauth', 'system-auth', and 'su'.

The first one found determines the PAM service name used by the `vovpamauth`.

```
Example of a vovauth file. Only the auth part is required.
auth sufficient pam_rootok.so
auth required pam_opendirectory.so
```

## Attribute Value Stream (AVS) File Syntax

Attribute Value Stream (AVS) is a file format and data interchange format that uses human-readable text to describe nested objects and arrays of objects. This file format is similar to JSON (Javascript Object Notation).

### AVS Syntax

Characteristics of AVS:

- Data is in attribute/value pairs separated by "=".
- Data is separated by commas.
- Curly braces hold objects.
- Square brackets hold arrays.

- Attribute names are not quoted and are case-sensitive.
- Comments start with the "#" character and extend to the end of the line.

Here is an example of a simple AVS object containing two attribute/value pairs:

```
{ name = "triangle", sides = 3 }
```

AVS objects are often nested. Here's an example of an AVS object containing an array of polygon objects, showing a valid AVS comment line:

```
{ polygon_array = [
 # the following are array elements
 { name = "triangle", sides = 3},
 { name = "rectangle", sides = 4},
 { name = "pentagon", sides = 5},
]
```

The preceding example illustrates that AVS allows an optional comma after the last attribute/value pair in an object or after the last object in an array.

## Object Definition

An AVS *object* is defined as a list of 0 or more attribute/value pairs separated by commas and wrapped by curly braces.

A properly formatted AVS configuration file contains one top-level object. Comments can precede, follow, or be embedded in the top-level AVS object. The values can contain nested objects and arrays.

## Attribute/Value Pairs

An attribute / value pair specification is formatted as follows. When no "= value" is specified, the attribute is given a null value. Null valued attributes formatted in this way are typically used for "flags" which impart meaning by their presence/absence in the object.

```
attribute = value
or
attribute
```

An *attribute* is a valid AVS *name*, which is an unquoted string containing any of the following characters:

- Upper or lower case letters or an underscore "\_" (required for initial character)
- Digits 0-9

A *value* may be any of the following types:

|               |                                                                                                                       |
|---------------|-----------------------------------------------------------------------------------------------------------------------|
| <b>Number</b> | Positive or negative 64 bit integers or floating point numbers.                                                       |
| <b>String</b> | Any characters surrounded by double quotes. If the string contains a double quote it is preceded by a "\"" character. |
| <b>Name</b>   | An unquoted string conforming to the attribute name rules mentioned above.                                            |
| <b>Object</b> | An AVS object (that is encased by curly braces).                                                                      |



**Array** An AVS array of objects, which is a list of AVS objects, surrounded by square brackets and separated by commas.

## Known Issues

### **Problem: on Windows 2000, the vovserver Generates WSAerror 10054**

This is a known problem in Windows 2000. See article [263823](#) in the Microsoft Knowledge Base, which recommends that you install Service Pack 2.

## Health Monitoring and vovnotifyd

You can set up basic Altair Accelerator health monitoring tests with mail notification. When the Accelerator system gets into any of your defined "unhealthy" conditions, a list of configured users will receive alert email notifications.

Tests are provided that check for the following conditions:

- Long jobs that are stuck: stuck jobs do not use any CPU
- Someone has jobs waiting in queue for too long
- Any user has an unusually high ratio of failed jobs
- Any host(s) fails all jobs
- The server size (and other server related parameters) is growing
- There are too many out of queue jobs (for Allocator (also known as MultiQueue) setup only)

The checks are all procedures of which the name starts with "doTestHealth". These checks are defined in one or more files of the following files:

| Role             | Location                                    | Notes                |
|------------------|---------------------------------------------|----------------------|
| Global           | \$VOVDIR/tcl/vtcl/<br>vovhealthlib.tcl      | Part of distribution |
| Site specific    | \$VOVDIR/local/<br>vovhealthlib.tcl         | Optional             |
| Project specific | PROJECT.swd/vovnotifyd/<br>vovhealthlib.tcl | Optional             |

### **Configuring Health Monitoring**

By default, all checks defined in the `vovhealthlib.tcl` files are enabled.

If you want to change parameters in the health checks, you need to change the `config.tcl` file in the `vovnotifyd` directory.

The following table contains an example of `vovnotifyd/config.tcl`:

```
set NOTIFYD(server) "tiger"
set NOTIFYD(port) 25
set NOTIFYD(sourcedomain) "mycompany.com"

set admin "dixin"
set cadmgr "john"

#
Check if we have long stuck jobs. Check every 1 minute. If we have such
jobs, send alert emails to the owner of the job (@USER@) and
admin (here "dixin").
#
Definition of "long stuck job": has been running at lease 10 hours and but has used
no
more than 20 seconds CPU time in total.
#
registerHealthCheck "doTestHealthLongJobsNoCpu -longJobDur 10h -minCpu 20" -
checkFreq 1m -mailFreq 1d -recipients "@USER@ $admin"

#
Check if we have jobs stuck, i.e., jobs that are not burning any CPU at all. And
this situation
has been persistent for at least 10 minutes.
#
Check every 1 minute. If we have such jobs, send alert emails to the owner of the
job (@USER@) and admin (here "dixin").
#
This check is similar to doTestHealthLongJobsNoCpu but will be quicker to detect
stuck jobs.
#
registerHealthCheck "doTestHealthJobStuck -maxNoCpuTime 10m" -checkFreq 1m -
mailFreq 1d -recipients "@USER@ $admin"

#
Check if any user has too many failed jobs. Check every 30 minutes. If we
have such users, send alert emails to the owner of the job (@USER@),
admin (here "dixin") and cadmgr (here "john").
#
Definition of "too many failures": a user has at least 1000 jobs in NetworkComputer
with at least 90% failures
#
registerHealthCheck "doTestHealthTooManyFailures -minJobs 1000 -failRatio 0.9" -
checkFreq 30m -mailFreq 1d -recipients "@USER@ $admin $cadmgr"

#
Check if the server size is growing. Check other server related parameters as
well, including number of jobs, number of queued jobs, etc.
#
For everything that is checked, if the number grows over 60.0% compared to last
time
it is checked, send alert emails to the admin (here "dixin")
and cadmgr (here "john").
#
Also send alert emails if the number of files is 5.0 times or more than the number
of jobs.
#
```

```
Check every 2 hours and send such alert emails once a day (1d).
#
registerHealthCheck "doTestHealthServerSize -filejobsRatio 5.0 -warnPercent 60.0" -
checkFreq 2h -mailFreq 1d -recipients "$admin $cadmgr"

#
Check if some user has jobs sitting in the queue for too long.
Check every 2 hours.
#
If we find such users, send alert emails to the user
and cadmgr (here "john"). Send such alert emails once a day (1d).
#
Definition of "waiting for too long": none of jobs in one category(bucket)
get dispatched in the last 4 hours.
#
#
registerHealthCheck "doTestHealthJobsWaitingForTooLong -maxQueueTime 4h" -checkFreq
2h -mailFreq 1d -recipients "@USER@ $cadmgr"
```

The config.tcl file is checked for updates at regular intervals controlled by the variable `NOTIFYD(timeout)`.

## Limits for Objects and Strings

### Numerical Limits

| Description    | Applies to | Default | Min | Max   | Policy Variable | Notes                                                                           |
|----------------|------------|---------|-----|-------|-----------------|---------------------------------------------------------------------------------|
| File tail name | Files      | 255     | -   | -     | -               | This is everything after the directory name                                     |
| Full path name | Files      | 1023    | 128 | 16000 | maxPathLengt    | Windows limit is 259, OSX is 1023, Linux is 1023 or 4095                        |
| NFS delay      | Files      | 0       | 0   | 300   | nfsdelay        | The number of seconds to wait before trusting the NFS cache for file properties |

| Description          | Applies to | Default  | Min | Max    | Policy Variable | Notes                                                              |
|----------------------|------------|----------|-----|--------|-----------------|--------------------------------------------------------------------|
| Time tolerance       | Files      | 0        | 0   | 600    | timeTolerance   | The max time skew between the server and any host used as a tasker |
| Child processes      | Jobs       | 40       | -   | -      | -               | This is the number that are tracked for job statistics             |
| Command length       | Jobs       | 40960    | 128 | 100000 | maxCommand      |                                                                    |
| Job array size       | Jobs       | 10000    | 10  | 100000 | maxJobArray     |                                                                    |
| Job class length     | Jobs       | No limit | -   | -      | -               | Limited by available memory only                                   |
| Job directory        | Jobs       | 1023     | -   | -      | -               | Limited by OS                                                      |
| Job environment      | Jobs       | 512      | 128 | 16000  | maxEnvLength    | Used named environments if more characters are needed              |
| Job legal exit codes | Jobs       | 100      | -   | -      | -               | Space-separated list                                               |
| Job name length      | Jobs       | No limit | -   | -      | -               | Limited by available memory only                                   |
| Job project length   | Jobs       | No limit | -   | -      | -               | Limited by available memory only                                   |
| Job resources        | Jobs       | 1024     | 128 | 16000  | maxResources    |                                                                    |

| Description           | Applies to | Default    | Min | Max     | Policy Variable | Notes                                                                                |
|-----------------------|------------|------------|-----|---------|-----------------|--------------------------------------------------------------------------------------|
| Resource map length   | Jobs       | 1024       | 512 | 64000   | resmap.max.n    |                                                                                      |
| Dependency levels     | Nodes      | 4095       | -   | -       | -               | This is the max number of dependency levels in a flow graph including files and jobs |
| Number of smart sets  | Sets       | 50         | 10  | 5000    | maxSmartSets    | Too many smart sets can result in reduced server performance                         |
| Selection rule        | Sets       | 511        | -   | -       | -               |                                                                                      |
| Set name              | Sets       | 511        | -   | -       | -               |                                                                                      |
| Property name length  | Properties | 255        | 16  | 1024    | prop.maxName    |                                                                                      |
| Property value length | Properties | 131071     | 128 | 1048576 | prop.maxStrin   |                                                                                      |
| Tcl variables         | Scripts    | 2147483647 | -   | -       | -               | Tcl variable values are limited to 2GB which may restrict some strings               |

## Character Type Restrictions

| Description | Applies to | Notes                                                       |
|-------------|------------|-------------------------------------------------------------|
| Job name    | Jobs       | Special characters and spaces are okay to use in job names. |

| Description             | Applies to | Notes                                                                                                                                                                                                                   |
|-------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Job command             | Jobs       | Most special characters are okay to use in commands. Redirects (e.g. > and <), pipes (e.g.  ), logical operators (e.g. && or   ) and semicolons need to be escaped unless they are meant to be used as shell operators. |
| Job substitutions       | Jobs       | @JOBID@, @NCJOBID@, and @IDINT@ in job names are substituted with the job's values for the job name, environment, and command.                                                                                          |
| Job array substitutions | Jobs       | @INDEX@, @ARRAYJOBID@, @ARRAYID@, and @ARRAYIDINT@ in jobs that are in job arrays are substituted with the job's values                                                                                                 |
| Set name                | Sets       | Special characters and spaces are okay to use in set names. Colons are used to denote hierarchy. Sets starting with "@@@" or "tmp:" are temporary sets and should not be used.                                          |

## VovScope

VovScope is a vovserver performance logging mode. When enabled, logging is activated for certain classes of vovserver activity including system calls and network communication to clients.

Start and stop VovScope logging as follows:

```
Start:
vovservermgr configure set_debug_flag VovScope

Stop:
vovservermgr configure reset_debug_flag VovScope
```

- VovScope logs vovserver interactions with clients and activities in the main dispatch loop.
- VovScope logs information about all clients (based on an open file descriptor), including vovresourced, vovdbd and user user CLI commands. The vovshow -clients can be used to find a complete list of VOV clients.
- VovScope logs the following activity:

- 1.** messages exchanged with clients – reads and writes
  - 2.** dispatch loop request handling—marked by “FD”
  - 3.** dispatch events – for example “GetProperty” and “StartUpNew”
- Segments of messages sent to or from vovserver are shown in the VovScope log. ASCII data and binary data are dumped in the log entries.

The profiling records are written in the logs folder under the server working directory. The profiling log files have filenames that start with prefix `vovscope*`.

# 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/](http://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

## Special Characters

-block option, proejct [94](#)  
.cshrc [17](#)  
.profile [17](#)  
@ARCHITECTURE@ [165](#)  
@CPUS@ [165](#)  
@DISPLAY@ [165](#)  
@HOST@ [165](#)  
@MACHINE@ [165](#)  
@MODEL@ [165](#)  
@MODELFULL@ [165](#)  
@OS.VERSION@ [165](#)  
@OS@ [165](#)  
@OSCLASS@ [165](#)  
@RAM@ [165](#)  
@RAMFREE@ [165](#)  
@RAMTOTAL@ [165](#)  
@RELEASE@ [165](#)  
@SMARTSUSPEND@ [165](#)  
@STD@ [165](#)  
@SWAP@ [165](#)  
@SWAPFREE@ [165](#)  
@SWAPTOTAL@ [165](#)  
@USER@ [165](#)  
@VIEW@ [165](#)  
@VOVARCH@ [165](#)  
/usr/tmp [152](#)

## A

access to help [24](#)  
acl.default.jobs.leader [48](#)  
add resources [234](#)  
administrator topics [241](#)  
advanced control of the product ports [33](#)  
advanced tasker topics [211](#)  
advanced taskers topics [211](#)  
alerts [108](#)  
alerts from liveness tasks [101](#)  
alerts, clear [108](#)  
alerts, manage [108](#)  
alerts, maximum number [108](#)  
alerts, Tcl API [108](#)  
alerts.max [48](#)  
allowcoredump [48](#)



- allowForeignJobsOnUserTaskers [48](#)
- allowUiForSecurityFile [48](#)
- Attribute Value Stream (AVS) file syntax [247](#)
- attributes of taskers [152](#)
- autoForgetFailed [48](#)
- autoForgetOthers [48](#)
- autoForgetRemoveLogs [48](#)
- autoForgetValid [48](#)
- autokill of jobs by vovtasker [159](#)
- autoLogout [48](#)
- automatic zipping and unzipping files [140](#)
- autoRescheduleCount [48](#)
- autoRescheduleOnNewHost [48](#)
- autoRescheduleThreshold [48](#)
- autoShutdown [48](#)
- autostart directory [100](#)
- AVS [247](#)

## B

- backup of trace [29](#)
- bin/bash [17](#)
- bin/csh [17](#)
- bin/tcsh [17](#)
- blackholedetection [48](#)
- blackholeDiscardTime [48](#)
- blackholeFailedJobs [48](#)
- blackholeFailRate [48](#)
- blackholeMaybeTime [48](#)
- blackholeSuspendTime [48](#)
- block a project from returning to shell [96](#)
- bps [184](#)
- bucketValidTaskerCountPeriod [48](#)

## C

- cache, equivalence [135](#)
- canonical file names [127](#)
- capacity in taskers [144](#)
- capacity, tasker [152](#)
- cgi.max [48](#)
- change host [94](#)
- change owner [94](#)
- change ownership of VOV projects [96](#), [96](#), [96](#)
- change the exclude.tcl file [139](#)
- change the project name [94](#), [94](#)
- change the server host [95](#)
- change user [94](#)

- checkoutHostLowerCase [48](#)
- checkoutUserLowerCase [48](#)
- choicemask resource [234](#)
- client/server architecture [6](#)
- clock synchronization [43](#)
- coefficient in tasker [152](#)
- coefficient of tasker [144](#)
- color of taskers [163](#)
- comm.buffer.compress.level [48](#)
- comm.buffer.compress.packetSizeThreshold [48](#)
- comm.buffer.compress.stringPercentThreshold [48](#)
- comm.maxBufSize [48](#)
- comm.maxTimeToCompleteFlush [48](#)
- comm.minTimeForCompulsoryFlush [48](#)
- configuration files for vovagent [193](#)
- configure a failover server replacement [122](#)
- configure the tls/ssl protocol [40](#)
- configuring job and license checkout matching [239](#)
- configuring user groups via the vovusergroup utility [118](#)
- conflict, port number [31](#)
- connect a single tasker [161](#)
- connect to a project [108](#), [108](#)
- connect to a third party batch processing system [184](#)
- consumable resources [222](#)
- corrupt trace, recovering [29](#)
- cpuprogressWindowSize [48](#)
- CPUs in a tasker [144](#)
- CPUs, in tasker [152](#)
- crash recovery mode [125](#)
- crash recovery restart [125](#)
- crashRecoveryMaxExtension [48](#)
- crashRecoveryPeriod [48](#)
- crashRecoveryQuietTime [48](#)
- create a VOV project [10](#)
- customize actions needed to enable access to Altair Accelerator products on Windows [21](#)

## D

- daemon, automatic start [100](#)
- daily.log [29](#)
- database, JOBSTATUS [132](#)
- database, LINK [130](#)
- databases [128](#)
- defaultStopSignalCascade [48](#)
- defaultStopSignalDelay [48](#)
- defaultSuspendSignalCascade [48](#)
- define equivalences for file names [133](#)

- delete resources [228](#)
- descriptors [44](#)
- destroy a project [121](#)
- disable handle matching [239](#)
- disablefileaccess [48](#)
- diskspacecheck.minfreemb [48](#)
- diskspacecheck.minfreepercent [48](#)
- dynamic resource map configuration [232](#)

## E

- emptyBucketAge [48](#)
- enable a project [102](#)
- enable Altair Accelerator for non-interactive shells [19](#)
- enable CLI access on UNIX [17](#)
- enable CLI access on Windows [20](#)
- enable the command prompt to communicate with a running product server [22](#)
- enable the shell to communicate with a running product server [19](#)
- enableLdap [48](#)
- enablePam [48](#)
- enableTimestampCheckAllFiles [48](#)
- enableTimestampCheckForRetrace [48](#)
- enableWaitReasons [48](#)
- enterpriselicense.burst [48](#)
- enterpriselicense.delay.decrease [48](#)
- enterpriselicense.delay.increase [48](#)
- epollbuf [48](#)
- equivalence cache [135](#)
- examples, job submission with triggers [104](#)
- exclude files from the graph [136](#)
- external webserver [39](#)

## F

- failover configuration, tips [123](#)
- failover server candidates [122](#)
- failover.maxdelaytovote [48](#)
- failover.sh [123](#)
- failover.usefailovertaskergrouponly [48](#)
- fairshare.allowAdminBypass [48](#)
- fairshare.default.weight [48](#)
- fairshare.default.window [48](#)
- fairshare.maxjobsperbucket [48](#)
- fairshare.maxjobsperloop [48](#)
- fairshare.overshoot.damping [48](#)
- fairshare.relative [48](#)
- fairshare.relative\_alpha [48](#)
- fairshare.updatePeriod [48](#)

- fairshareMode [48](#)
- file descriptors [44](#)
- file names [127](#)
- file system offsets [242](#)
- FILE, databases [128](#)
- files in the server working directory [29](#)
- files, equivalences, exclusions [127](#)
- FILEX, databases [128](#)
- forget unneeded resource maps [233](#)
- free disk space [167](#)

## G

- GLOB, databases [128](#)
- guest access port [38](#)

## H

- handle matching, disable [239](#)
- handling the overflow event [104](#)
- hardware resources [211](#)
- health monitoring and vovnotifyd [249](#)
- help, Accelerator [24](#)
- hog.protection.clientdelay [48](#)
- hog.protection.enable [48](#)
- hog.protection.jobcountthreshold [48](#)
- HOST\_OF\_jobId and HOST\_OF\_ANCECEDENT [215](#)
- host, change [139](#)
- how vovserver failover works [123](#)
- HTTP access models [38](#)
- httpSecure [48](#)

## I

- idrange [48](#)
- indirect taskers [144](#), [144](#)
- interface with Accelerator using vovelasticd [205](#)
- interface with LSF using taskerLSF.tcl [196](#)
- interface with LSF using vovlsfd [197](#)
- interfaces to Accelerator, SGE, LSF, and others [184](#)
- interfaces to other batch processing systems [190](#)
- interfacing FlowTracer with Accelerator [186](#)
- internal webserver [39](#)

## J

- jetstreams.debug [48](#)
- jetstreams.enable [48](#)
- jetstreams.threshold [48](#)

- job resources [222](#)
- job status triggers [104](#)
- jobQueuePolicy [48](#)
- JOBSTATUS database [132](#)
- JOBSTATUS, databases [128](#)
- journal files [125](#)
- journals [97](#)

## K

- known issues [249](#)

## L

- la.adjustForUnconsumed.enable [48](#)
- la.adjustForUnconsumed.timeWindow [48](#)
- la.feature.expirationThreshold [48](#)
- la.ooq.waves.disable [48](#)
- legacy webserver [38](#)
- license-based resources [235](#)
- limits [44](#)
- limits for objects and strings [251](#)
- LINK database [130](#)
- LINK, databases [128](#)
- list projects in the registry [15](#)
- liverecorder.logdir [48](#)
- liverecorder.logsize [48](#)
- Immgr stop [125](#)
- load sensor [152](#)
- load, in tasker [152](#)
- local resource maps [225](#)
- location and structure [28](#)
- log files [29](#), [97](#)
- log.nodestatus.changedetail [48](#)
- log.nodestatus.loggingtype [48](#)
- logfile, tasker [152](#)
- logical file names [127](#)
- logs,checkouts,compress [48](#)
- logs,checkouts,handler [48](#)
- logs,jobs,compress [48](#)
- logs,jobs,handler [48](#)
- logs,journals,compress [48](#)
- logs,journals,handler [48](#)
- logs,resources,compress [48](#)
- logs,resources,handler [48](#)
- logs,server,compress [48](#)
- logs,server,handler [48](#)
- logs,waitreasons,compress [48](#)

logs,waitreasons,handler [48](#)  
longServiceWarningDuration [48](#)

## M

manage reservations [172](#)  
map host names [220](#)  
max idle time of tasker [144](#)  
max lifetime of tasker [144](#)  
max load of tasker [144](#)  
max number of jobs a tasker can execute [144](#)  
maxAgeRecentJobs [48](#)  
maxBufferSize [48](#)  
maxChildren [48](#)  
maxCommandLineLength [48](#)  
maxEnvLength [48](#)  
maxeventqueue [48](#)  
maxfile, vovagent.cfg [193](#)  
maxidle, vovagent.cfg [193](#)  
maxJobArray [48](#)  
maxLevel [48](#)  
maxlife, vovagent.cfg [193](#)  
maxNormalClients [48](#)  
maxNotifyBufferSize [48](#)  
maxNotifyClients [48](#)  
maxORsInResourceExpressions [48](#)  
maxPathLength [48](#)  
maxQueueLength [48](#)  
maxResMap [48](#)  
maxResourcesLength [48](#)  
maxSmartSets [48](#)  
message, vovtasker [144](#)  
method 1: use windows explorer to set command line environment [20](#)  
method 2: using Windows command prompt to set command line environment [21](#)  
metrics.autoSavePeriod [48](#)  
metrics.enable [48](#)  
metrics.keepFilesPeriod [48](#)  
metrics.maxmem [48](#)  
minhw [48](#)  
minimum free disk [144](#)  
minMemFreeOnServer [48](#)  
minSwapFreeOnServer [48](#)  
missing registry entry files [16](#)  
monitor LSF jobs in the GUI [204](#)  
monitor taskers [164](#)

## N

- ncmgr stop [125](#)
- negated reservations [172](#)
- negated resources [222](#)
- netInfo [48](#)
- network conditioning [241](#)
- network overhead [246](#)
- NFS and VOV [243](#)
- nfsdelay [48](#)
- notifyMaxEffort [48](#)
- notifySkip [48](#)
- number of reservations and system performance [172](#)

## O

- online help [24](#)
- operations by security level [115](#)
- owner [94](#)
- owner, change [139](#)

## P

- PDF, access [24](#)
- persistant reservations [172](#)
- PHANTOM, databases [128](#)
- policies and resources [233](#)
- port conflict [31](#)
- port number [31](#)
- predefined vovresources:: procedures [169](#)
- preemption.load [48](#)
- preemption.log.allrules [48](#)
- preemption.log.verbosity [48](#)
- preemption.max.time.overall [48](#)
- preemption.max.time.rule [48](#)
- preemption.module [48](#)
- preemption.rule.cooldown [48](#)
- preemptionFairshare [48](#)
- preemptionPeriod [48](#)
- preemptionResourceRelease [48](#)
- processRetentionTime [48](#)
- project host, change [94](#), [139](#)
- prop.maxNameSize [48](#)
- prop.maxStringSize [48](#)
- protection from DoS attacks [117](#)

## Q

- quantitative resources [222](#)

## R

- RAM, monitor [165](#)
- rank the resource agents [228](#)
- readonlyport [48](#)
- recover trace [29](#)
- refresh rate [230](#)
- registry file [96](#)
- registry concerns [15](#)
- rehost a project [95](#)
- rehost project [139](#)
- rejectIndirectTaskerJobs [48](#)
- relationship between FlowTracer, Accelerator, and Monitor [8](#)
- remote shell command [144](#)
- request hardware resources [211](#)
- resmap.max.map.length [48](#)
- resmap.max.maps.in.expression [48](#)
- resmap.max.nru.time [48](#)
- resmap.sw.types [48](#)
- resMapGroupDefault [48](#)
- resMapToolDefault [48](#)
- resMapUserDefault [48](#)
- resource daemon configuration [230](#)
- resource management [222](#)
- resource map expiration [228](#)
- resource map reservation [232](#)
- resource map set [224](#)
- resource mapping [225](#)
- resource monitoring [224](#)
- resource reservation [227](#)
- resources.log [29](#)
- restart, crash recovery [125](#)
- resuserDisableMatchingThreshold [48](#)
- resuserEnableNRUMatches [48](#)
- resusermatchtolerance [48](#)
- resusermaxmatches [48](#)
- run periodic tasks with vovcrontab [102](#)
- run periodic tasks with vovliveness [101](#)
- run vovtsd as a different user [219](#)
- runAsWX [48](#)

## S

- safe server threads [46](#)
- sanity check for vovserver [106](#)
- sanity,smartsets [48](#)
- saveToDiskPeriod [48](#)
- sched.busy.thresh [48](#)



- [sched.maxpostponedjobs 48](#)
- [sched.megapoll.threshold 48](#)
- [sched.taskerstats.enable 48](#)
- [schedMaxEffort 48](#)
- [security 112](#)
- [security levels 112](#)
- [server candidates 122](#)
- [server capacity 44](#)
- [server configuration 37](#)
- [server configuration parameters 48](#)
- [server configuration variables 37](#)
- [server working directories subdirectories 29](#)
- [server working directory 28](#)
- [server working directory name 31](#)
- [servercandidates.tcl 122, 123](#)
- [serverinfo.tcl 123](#)
- [set the range for VovId 47](#)
- [setup.tcl 123](#)
- [shutting down a project 120](#)
- [ssh setup 241](#)
- [ssl.enable 48](#)
- [ssl.enableRawUrlOnHttp 48](#)
- [start a remote UNIX tasker details 179](#)
- [start a remote vovtasker with vovtsd 219](#)
- [start and configure the server 26, 99](#)
- [start the vovserver 26](#)
- [starting vovresourced 230](#)
- [stop taskers 180](#)
- [stop the server 120](#)
- [subdirectories, server working directory 29](#)
- [subordinate resources 195](#)
- [substituteArrayIdInResources 48](#)
- [substituteArrayIndexInResources 48](#)
- [substituteJobIdInResources 48](#)
- [swap, monitor 165](#)
- [switchToExtBufThresh 48](#)
- [symbolic resources for tasker 165](#)

## T

- [tasker attributes 152](#)
- [tasker capabilities 144](#)
- [tasker color 163](#)
- [tasker configuration parameters 156](#)
- [tasker ID 152](#)
- [tasker name 144](#)
- [tasker owner 178](#)

- tasker policies [172](#)
- tasker power [158](#)
- tasker RAM saturation [160](#)
- tasker resources [165](#)
- tasker status [163](#)
- tasker.childProcessCleanup [48](#)
- tasker.max.reserve [48](#)
- tasker.maxWaitToReconnect [48](#)
- tasker.props.enable [48](#)
- taskerautokill [159](#)
- taskerheartbeat [48](#)
- taskers introduction [142](#)
- taskers running as root: security implications [180](#)
- taskers: mixed Windows NT and UNIX environment [179](#)
- taskers: vtk\_tasker\_set\_timeleft [216](#)
- taskers.tcl file [143](#), [172](#)
- tasksMaxEffort [48](#)
- the taskerClass.table file [167](#)
- thin clients [111](#)
- thread.preempt.max [48](#)
- thread.service.enable.fsgroup [48](#)
- thread.service.enable.query [48](#)
- thread.service.fork.ref.time [48](#)
- thread.service.max [48](#)
- thread.service.min.nodes [48](#)
- thread.service.min.resmaps [48](#)
- thread.service.min.taskers [48](#)
- time tolerance [43](#)
- time variant taskers [167](#)
- timestamp comparison [43](#)
- timeTolerance [48](#)
- tools.log [29](#)
- trace, backup [29](#)
- trace.db [29](#)
- trigger events [104](#)
- TRIGGER property [104](#)
- troubleshoot taskers [217](#)
- troubleshooting the server [93](#)
- troubleshooting the UNIX setup [20](#)
- troubleshooting: cannot enable project [111](#)
- trustHostReportedByClient [48](#)
- types of vovtasker binaries [142](#)

## U

- uncountable resources [222](#)
- use Accelerator help [24](#)

- use vovequiv to check equivalences [136](#)
- usepoll [48](#)
- user interfaces [23](#)
- userGroupLowerCase [48](#)
- userLowerCase [48](#)
- using PAM (Pluggable Authentication Modules) for browser authentication [246](#)
- using vovcontrab [102](#)
- usrtmp [48](#)

## V

- variable, server configuration [37](#)
- verify access to Altair Accelerator products [18](#)
- verify context is working [22](#)
- VOV [47](#)
- VOV project server host [9](#)
- VOV projects [9](#)
- VOV protocol summary [42](#)
- VOV registry [14](#)
- VOV Subsystem Administrator Guide [5](#)
- VOV\_ENV\_DIR [29](#)
- VOV\_HOST\_NAME [44](#), [161](#)
- VOV\_PORT\_NUMBER [31](#)
- VOV\_PROJECT\_NAME [29](#), [94](#), [161](#)
- VOV\_READONLY\_PORT\_NUMBER [31](#)
- VOV\_STDOUT\_SPEC [24](#)
- VOV\_SWD\_KEY [111](#)
- vovagent [193](#)
- vovagentcfg [193](#)
- VOVARCH [6](#), [165](#)
- vovautostart [100](#), [122](#)
- vovbrowser [24](#)
- vovbuild [24](#)
- vovcache [111](#)
- vovcontrab [102](#)
- vovdb [144](#)
- vovdoc [24](#), [24](#)
- vovequiv [135](#)
- VOVEQUIV\_CACHE\_FILE [111](#), [133](#)
- vovforget [139](#)
- vovid [24](#)
- VovId [47](#)
- vovIdHigh [48](#)
- vovIdLow [48](#)
- vovinfo [111](#)
- vovnotifyd [29](#)
- vovpreemptd [29](#)

vovproject [26](#), [106](#), [123](#)  
vovproject backup [29](#)  
vovproject restore [29](#)  
vovresourced [106](#), [228](#), [230](#)  
vovresourcemgr [231](#)  
vovresSetFlags [239](#)  
vovrestoretrace [29](#)  
vovretraced [125](#)  
vovscope [254](#)  
vovserver [24](#), [27](#), [106](#), [239](#)  
vovserver autostart [100](#)  
vovserver host [139](#)  
vovserverdir [122](#)  
vovservermgr [35](#)  
vovservsel.tcl [123](#)  
vovset [139](#)  
VOVSETS, databases [128](#)  
vovsh [47](#)  
VOVSTOP [159](#)  
vovstring.maxLength [48](#)  
vovtasker [123](#)  
vovtasker states [142](#)  
vovtaskermgr [106](#), [147](#)  
vovtaskermgr reserve [172](#)  
vovtaskers [122](#)  
vovtriggerd [104](#)  
vovtsd [218](#)  
vovtsd (taskers service daemon [218](#)  
vovtsd on UNIX [220](#)  
vovusergroups [118](#)  
vovusergroups support [118](#)  
vovzip [140](#)  
vovzip\_host [140](#)  
vovzipdir [140](#)  
vtk\_equivalence [133](#)  
vtk\_equivalence\_get\_cache [133](#)  
vtk\_equivalence\_set\_cache [133](#)  
vtk\_exclude\_rule [139](#)  
vtk\_generic\_get [31](#), [47](#)  
vtk\_job\_control [182](#)  
vtk\_place\_find [140](#)  
vtk\_place\_get [140](#)  
vtk\_place\_set [140](#)  
vtk\_port\_number [31](#)  
vtk\_reservation\_create [172](#)  
vtk\_reservation\_get [172](#)  
vtk\_resourcemap\_delete [228](#)

vtk\_resourcemap\_forget [228](#)  
vtk\_resourcemap\_set [140](#), [239](#)  
vtk\_server\_config [47](#)  
vtk\_tasker\_config [159](#)  
vtk\_tasker\_define [144](#)  
vtk\_tasker\_reserve [172](#)  
vtk\_tasker\_set\_default [144](#)

## **W**

web server configuration [38](#)  
webport [48](#)  
workaround for misspelled resource [232](#)  
wx.dispwithbucketres [48](#)

## **Z**

zip, automatic [140](#)  
ZIP, databases [128](#)