



ALTAIR

ONLY FORWARD

Altair Accelerator 2024.1.0

User Guide

Contents

Altair Accelerator User Guide	4
Use Accelerator Help.....	7
Accelerator Quick Start.....	9
Command Line Interface.....	12
Quick Reference.....	14
User Shell Setup.....	16
Verify Your Setup.....	17
Running a Job.....	18
Submit Jobs.....	18
View Job Progress and Results.....	27
Manage Jobs.....	32
Rerun Jobs.....	37
Job Management.....	40
Job Arrays.....	40
Jobname Attribute.....	40
Jobclasses.....	41
Pre-Command and Post-Command Job Conditions.....	42
Schedule Job Submission.....	44
Queue Selection.....	44
Scheduled Jobs.....	46
Priority.....	47
Distributed Parallel Support.....	49
Multiphase Support.....	54
Job Submission Arguments.....	56
Modify Running Jobs.....	57
Interactive Jobs.....	57
Modify Scheduled Jobs.....	59
Restrictions and Consequences.....	61
FairShare Groups.....	63
Job Placement Policies.....	64
Clean Up Log Files.....	66
Debug Jobs without Running Accelerator.....	68
Job Runtime - Monitor and Profile.....	70
Job I/O Profiling.....	73
Monitor the Workload.....	75
List Jobs.....	75
Summary of All Jobs.....	77
Notification of Job Status.....	78
Invoke the GUI.....	79
Icons.....	80
Show the Hosts/Taskers.....	82

Monitor Jobs, Taskers and Resources.....	88
Statistical Information about Resources and Jobs.....	90
Statistics.....	90
Resource Statistics.....	90
Resource Plots.....	92
Job Resource Plots.....	95
Environment Control.....	98
Select a Named Environment.....	99
Use Snapshot with Named Environment.....	99
Job Resources.....	101
Cross-platform Job Runs.....	101
Accelerator and ClearCase: nc run -clearcase.....	101
Request a License Before Executing Jobs.....	102
NUMA Control and CPU Affinity.....	104
CGROUPS for Jobs.....	105
Use Containers for Jobs.....	107
Match Jobs to Handles.....	108
Resource/Handle Matching.....	110
Advanced Information.....	111
Set the Range for VovId.....	111
Node Fields.....	112
Formatting Strings.....	123
Time Specifications.....	124
Selection Rules.....	126
Migrate from LSF.....	129
Frequently Asked Questions and Troubleshooting Tips.....	131
HPC Advice.....	134
NVIDIA™ GPUs Support in Accelerator.....	136
Simulation Scripts.....	138
Sanity Check for vovserver.....	139
Disable Regular User Login.....	140
Auxiliary Group Membership.....	142
Troubleshooting.....	143
Deprecated Commands.....	146
LSF Emulation.....	147
Legal Notices	151
Intellectual Property Rights Notice.....	152
Technical Support.....	156
Index	158

This guide describes basic tasks in Accelerator, including submitting jobs, tracking job information, and analyzing and solving common problems.

This chapter covers the following:

- [Use Accelerator Help](#) (p. 7)
- [Accelerator Quick Start](#) (p. 9)
- [Command Line Interface](#) (p. 12)
- [Quick Reference](#) (p. 14)
- [User Shell Setup](#) (p. 16)
- [Running a Job](#) (p. 18)
- [Job Management](#) (p. 40)
- [Schedule Job Submission](#) (p. 44)
- [Modify Running Jobs](#) (p. 57)
- [FairShare Groups](#) (p. 63)
- [Job Placement Policies](#) (p. 64)
- [Clean Up Log Files](#) (p. 66)
- [Debug Jobs without Running Accelerator](#) (p. 68)
- [Job Runtime - Monitor and Profile](#) (p. 70)
- [Monitor the Workload](#) (p. 75)
- [Statistical Information about Resources and Jobs](#) (p. 90)
- [Environment Control](#) (p. 98)
- [Job Resources](#) (p. 101)
- [Advanced Information](#) (p. 111)
- [Frequently Asked Questions and Troubleshooting Tips](#) (p. 131)
- [Deprecated Commands](#) (p. 146)
- [LSF Emulation](#) (p. 147)

 **Note:**

The terminology in this release has changed from the previous one.

The Accelerator products are built on platform called vov using a client-server architecture with remote-procedure-calls (RPC). The server software module is called vovserver. It communicates to clients using the vov protocol; vovservers can also be configured to respond to http requests: the REST API is implemented on top of http. There are several different client types, those that make requests to the vovserver are typically implemented using vovsh (the vov shell - a Tcl interpreter); those that respond to vovserver requests to run jobs or tasks are taskers and the software here is called vovtasker. The vovtasker can run on the same host as the vovserver or on a separate host; these hosts are typically referred to as compute nodes, compute hosts or execution hosts.

The architecture allows for multiple vovservers to communicate with each other via a vovagent. Examples of vovagents include vovwxd, indirect taskers and vovlad.

In the 2021.1.0 release, the term *slave* has been deprecated and has been replaced with the term *tasker*. The web user interface and the online documentation have been updated to reflect this change, as has the majority of the code base. Subsequent releases will complete the transition.

Accessing Accelerator

Accelerator can be accessed via the following media:

- **Web UI.** Configuring Accelerator properties, and viewing job status, configurations, available resources and more is available through the Web user interface.
- **GUI.** Graphical user interface, independent of the Web is also available for graphical views of current job and resource statuses.
- **CLI Command.** Commands are also available for configuration, viewing the status of jobs and resources. GUI and WebUI can be invoked through CLI commands.

Theory of Operation

During the initial setup, the Accelerator host server, vovserver, establishes a main port for communication and additional ports for web access and read-only access. Afterwards, the vovserver waits for and responds to incoming connection requests from clients.

Clients consist of regular *clients* that request a particular service, *taskers* (server farms) that provide resources, and *notify* clients that listen for events. In addition to tasker-based resources, some clients provide central resources, which are stored in and tracked by the vovserver.

Regular clients can define jobs, or query data about jobs or system status. When a job is defined, it is normally placed in a scheduled state. Scheduled jobs are sorted into buckets. Jobs that have the same characteristics go in the same bucket. Buckets are placed in prioritized order for dispatching. This prioritization is based on FairShare, an allocation system. The top priority job in each bucket is dispatched when each of the defined resources (requests) for that job is available. The job requests can be fulfilled from the central pool as well as the tasker resources. When a tasker is found that completes the job's resource request, the job is dispatched to that tasker and the job status changes to running.

When the job has completed, the tasker notifies the vovserver. The resources, both tasker-based and central, are recovered, which allows subsequent jobs (queued in the buckets) to be dispatched. When completed, the job status is normally updated to either valid or failed.

As previously stated, in addition to dispatching jobs and processing their statuses, the vovserver responds to queries about system and job requests, publishes events to notify clients, and continues to process incoming job requests.

Known Limitations

In the Windows environment, PowerShell is not supported; it is strongly recommend to avoid using PowerShell.

Use Accelerator Help

Accelerator documentation is available in HTML and PDF format.

Access the Help when Accelerator is Running

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

```
% Accelerator cmd vovbrowser  
http://comet:6271/project
```

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

- vovshow -env RX** Displays the environment variables that match the regular expression RX provided.
- vovshow -fields** Shows the fields known to the version of VOV in use.
- vovshow -failcodes** 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.

Accelerator Quick Start

Accelerator has two main commands, `nc` and `ncmgr`.

- `nc` is used to submit, query, and stop jobs. This command can also be invoked as `vnc`.
- `ncmgr` is used to start a queue: `ncmgr start`. By default, the queue (`vnc`) starts in a server working directory (SWD) that is a subdirectory in `$VOVDIR/.../.../vnc`.

The output of `ncmgr start/stop` is logged in `/${VOVDIR}/local/logs/nc`, if it exists.

This page shows the usage messages that are generated by the `nc` and `ncmgr` commands.

nc

```
vnc: Usage Message
Usage: nc [-q queueName] <command> [command options]

Queue selection:
  The default queue is called "vnc".

You can specify a different queue with the option -q <queueName>
or by setting the environment variable NC_QUEUE.

Commands:
clean          Cleanup log files and env files.
debug         Show how to run the same job without Accelerator.
dispatch     Force dispatch of a job to a specific tasker.
forget       Forget old jobs from the system.
getfield     Get a field for a job.
gui          Start a simple graphical interface.
help        This help message.
hosts       Show farm hosts (also called taskers).
info       Get information about a job and its outputs.
list      List the jobs in the system.
jobclass  List the available job classes.
kerberos  Interface to Kerberos (experimental).
modify    Modify attributes of scheduled jobs.
monitor   Monitor network activity.
rerun     Rerun a job already known to the system.
resources Shows resource list and current statistics.
resume   Resume a job previously suspended.
run <job> Run a new job (also called 'submit').
preempt  Preempt a job.
stop     Stop jobs.
submit <job> Same as 'run'.
summary  Get a summary report for all my jobs.
suspend  Suspend the execution of a job.
taskerlist Show available tasker lists.
wait     Wait for a job to complete.
who      Report on who is using the system.
why      Analyze job status reasons.

Unique abbreviations for commands are accepted.

Advanced features:
  cmd <command>      Execute an arbitrary VOV command in the
```

```
context of the $product server.
source <file.tcl> Source the given Tcl file.
- Accept commands from stdin.

For more help type:
% $::command <command> -h

Copyright (c) 1998-2021, Altair Engineering.
```

ncmgr

This program manages the vovserver for Accelerator.

Usage

```
vncmgr: Usage Message

This program manages the vovserver for Accelerator.
Copyright (c) 1998-2022, Altair Engineering.

USAGE:
ncmgr help|info|rehost|reset|start|stop|cm [OPTIONS]

ACTIONS:
info [-queue|-q <name>] [-v]
reset [-soft | -hard | -h ]
rehost [-force] [-queue|-q <name>] -host <host>
start [-dir <server_working_dir>] [-force] [-queue|-q <name>]
[-port <port> ] [-webport <port>] [-roport <port>]
[-dbhost <host>] [-dbroot <path>] [-dbport <port>]
[-prod nc|wx|he] [-basequeue <name>] [-dd]
The default <server_working_dir> is
<...>/vnc.
This is the parent of the configuration (.swd) directory for
the queue.
stop [-force] [-freeze] [-freeze_nocpr] [-queue|-q <name>]
[-writeprdir <dirname>]
-force Do not prompt for confirmation
-freeze Instruct taskers to keep running and wait for a
new server
-freeze_nocpr Instruct taskers to keep running and wait for a
new server, and do not compress PR file
-writeprdir Writes the PR file to the specified directory
(which is created if necessary)
cm [-queue|-q name] <ACTION> [ARGUMENTS]
Configuration Management. Pass "help" for detailed usage.

EXAMPLES:
% ncmgr
% ncmgr -h
% ncmgr start -queue vnc2
% ncmgr start -port 6699 -queue vnc99
% ncmgr info
% ncmgr reset -soft
% ncmgr reset -hard
% ncmgr cm help
```

EXAMPLE TO STOP AND RESTART SERVER:

```
% ncmgr stop -freeze  
% ncmgr start -force  
% ncmgr stop -freeze -force -writeprdir /tmp/abc123
```

Command Line Interface

All user commands have the following structure:

```
% nc [-q qname] subcommand [options]
```

The command plus the subcommand is one of the following:

- nc clean
- cmd
- nc debug
- dispatch
- nc forget
- nc gui
- help
- nc hosts
- nc info
- nc list
- nc jobclass
- nc modify
- monitor
- nc rerun
- resume
- nc run
- source
- nc stop
- nc summary
- suspend
- nc wait

For example:

```
% nc help  
% nc run sleep 10  
% nc list  
% nc forget -mine
```

The Exclamation Point (!) Special Operator

Some Accelerator subcommands accept a single exclamation point, and interpret it to mean 'most-recent job run in the current directory'. This is meant for interactive use to avoid typing or copying the nine digit job ID.

This is not recommend for use in scripts, because it involves a scan of the jobs in the system. Instead, save the job ID returned when submitting the job and use the ID in queries.

The Accelerator subcommands that support this are:

- info
- getfield
- rerun

For example:

```
% nc info !
```

```
% nc info -l !
```

Some VOV commands that support this may sometimes be useful in Accelerator context, by preceding them with `nc cmd`:

- `vovset`
- `vovfire`
- `vsx, vsy`

Any unique prefix for the subcommand is accepted, which allows abbreviated forms of commands to be used. For example:

```
% nc l  
% nc li  
% nc lis  
% nc list
```

Quick Reference

Common Commands

help	Getting Accelerator help.
nc forget	Remove jobs from Accelerator queue.
nc getfield	Get detailed information on a job.
nc info	Get information on jobs.
nc list	Get a formatted list of jobs.
nc modify	Modifying jobs in the system.
monitor	Monitoring jobs and tasker machines.
notify	Email notification.
policy	Setting policy.
nc rerun	Re-running jobs.
nc run	Submitting jobs in Accelerator.
interactive	Running interactive jobs in Accelerator.
status	Getting status info in Accelerator.
nc stop	Stopping jobs in Accelerator.

Information Pages

Altair Accelerator User Guide	Introduction to Accelerator.
Installation Guide	Installation of Accelerator.
Manage	Managing Accelerator.
Test	Testing Accelerator after installation.
Troubleshoot	Troubleshooting Accelerator.

Administrative

Advanced Information	Advanced command usage.
Clean Up Log Files	Clean up log files.
Cross-platform	Cross-platform runs.

Environment Control	Controlling the VNC environment.
FairShare Groups	Setting up groups.
vnc_policy.tcl	VOV policy setup file.
Resource Management	Accelerator resource management.
Scheduled Jobs	Accelerator job queue.

User Shell Setup

To set up your user shell with Accelerator you need to know where the Altair Accelerator software has been installed.

Ask your system administrator.

User Setup: C-Shell, TCSH

Choose one of two choices:

- a) Modify your `~/ .cshrc` file directly by adding the following line:

```
# Add this to your .cshrc
source /<installation_directory>/<version>/<platform>/etc/vovrc.csh
```

- b) Run the `vovsetupuser` script, which creates a `~/ .vovrc` file and modifies your `~/ .cshrc` file to source the `~/ .vovrc` file.

```
% cd <installation_directory>/<version>/<platform>/
% cd scripts
% ./vovsetupuser -csh
```

User Setup: Bourne Shell, K-Shell, Z-Shell, Bash

Choose one of two choices:

- a) Source the file `$VOVDIR/etc/vovrc.sh`. It is recommended that you add the following line to your `~/ .profile` file:

```
# Add this to your .profile or .bashrc
. <installation_directory>/<version>/<platform>/etc/vovrc.sh
```

- b) Run the `vovsetupuser` script.

```
% cd <installation_directory>/<version>/<platform>/
% cd scripts
% ./vovsetupuser -sh
```

User Setup: Windows Command Shell

If Accelerator is installed in directory `R:\opt\altair\vov\2023.1.1`, you can set up your cmd shell by executing:

```
c:> R:\altair\vov\2023.1.1\win64\bat\vovinit.bat
```


Verify Your Setup

1. Run the following Altair Accelerator command to verify that your setup works:

```
% vovarch  
linux64
```


2. Run the command `vovversion` to show the version of Accelerator that is installed.

```
% vovversion  
2023.1.1
```

Running a Job

This section summarizes how to submit and run a job.


Information about submitting jobs will be covered in the sections [Submit Jobs with CLI Commands](#) and [Submit Jobs from the Browser](#).

 **Note:** To run a job, the working environment must be set. For information, refer to [User Shell Setup](#).


Submit Jobs

Submit Jobs with CLI Commands

This chapter provides examples of submitting jobs using CLI (command line interface) commands.

 **Note:** CLI commands are case insensitive. For example, `timetolerance` and `timeTolerance` represent the same command.

nc run

 **Note:** `nc modify -res` now support binary unit conversion for all memory based resources as a convenience from Petabytes (PB), TerabyteS (TB), or GigabyteS (GB) to Megabytes (MB), which is still used internally and reported by all commands. The input conversion will accept either decimal or integer form and are all case-insensitive, so for example both `nc run -r SWAP/1GB - sleep 0`, and `nc run -r RAM/0.1Tb - sleep 0` are supported.

The currently supported parameter names for which this conversion is supported are `RAM/`, `RAM#`, `RAMFREE#`, `RAMFREE/`, `RAMTOTAL#`, `RAMFREE/`, `SWAP/`, `SWAP#`, `SWAPFREE#`, `SWAPFREE/`, `SWAPTOTAL#`, `SWAPTOTAL/` and `TMP#` or `TMP/`. By default the unit is MB (Megabytes), where 1MB is $1 \ll 20$ bytes.

```
nc: Usage Message
```

```
NC RUN:
```

```
Run one or more jobs. The jobs are added to the system, and  
will remain in the system until you use 'forget' to forget them  
or they are automatically forgotten by the system.  
If taskers and resources are available, the jobs are dispatched  
immediately, else they are queued.
```

```
USAGE:
```

```
% nc run [OPTIONS] command ...
```

GENERAL OPTIONS:

```
-h                -- Help usage message.
-v <level>       -- Verbose level from 0 (silent) to 9 (very verbose).
--              -- Null option. In case of ambiguity, use this to separate
                  the options from the command.
```

In addition, the value of the environment variable NC_RUN_ARGS is prepended to the argument list for this command, while the value of NC_RUN_ARGS_AFTER is appended.

JOB CHARACTERISTICS OPTIONS:

```
-autokill <time> -- Kill job if it runs longer than specified time.
                  Set it to zero to disable autokill (the default).
-clearcase       -- This is a job to be run in a ClearCase view
                  (see docs).
-C <class>       -- The job belongs to the given class.
                  If argument is empty, the option is ignored.
                  Option can be repeated. The jobclass of the job
                  will be the last one specified.
-e <env>         -- Set the environment. Default is current env, as
                  defined by the variable VOV_ENV.
                  Setting this to the null string "" or to
                  "SNAPSHOT", forces the use of an environment
                  snapshot.
-e+ <env>        -- Append to existing environment.
-ep             -- Capture environment in a SNAPSHOT property. Uses
                  SNAPPROP environment.
-first          -- Schedule job first in its bucket.
-forceterm      -- In the case of interactive jobs where the output is
                  piped, the job's TERM environment variable is set
                  to 'network'. This option disables that behavior.
-fstokens <N>   -- Multiply weight of this job in FairShare by N.
                  Default 1, range is [0..50000]
-g <group>       -- Specify the FairShare group.
                  The .user:subgroup suffix will be added. You need
                  attach permission to run in the specified group.
-G <group.tag>   -- Specify the complete FairShare group with the
                  <group>.<tag> and/or <group>.<tag>:<subgroup> syntax.
                  (<tag> is typically a user.) If the <group> or
                  <subgroup> does not exist, it will be created with
                  the current user as the owner. You need attach
                  permission to run in the specified group.
-ioprofile      -- Activate enhanced job profiling.
-I, -Ir         -- Run interactive job. TTY signals like <ctrl>C are
                  propagated to the job. If the environment variable
                  VOV_INTERACTIVE_PING is set, its value (TIMESPEC
                  format. minimum is 1m) will be used to keep the
                  connection with the interactive job alive by pinging
                  the job at the specified interval.
-Il            -- Run interactive job. TTY signals like <ctrl>C are
                  kept local, not propagated to the job. Appropriate
                  for piping stdout to a file or command. See above
                  for usage of VOV_INTERACTIVE_PING.
-Ix            -- Run X Window based interactive job, no TTY, no wait.
                  Adds env D(DISPLAY=...) so job displays on submission
                  display. See above for usage of VOV_INTERACTIVE_PING.
-jobproj <name> -- The job project is set to <name>. The default is
                  determined by the environment variables VOV_JOBPROJ,
                  LM_PROJECT and RLM_PROJECT.
-jpp <JPP>      -- Specify a job-placement-policy. These policies are
                  advisory only. Legal values for JPP are a comma
```

separated list of one or more words from the following list:

At most one of these:

fastest -- This is the default job placement policy: among all the taskers that can execute a job, choose the one with the highest power. The assumption is that a tasker with a higher power will complete the job faster.

slowest -- Among all the taskers that can execute a job, choose the with the least power. This policy may be useful to run regression jobs on older, less powerful hardware.

first -- As soon as the scheduler finds a tasker to execute the job, it uses that tasker without checking all other taskers. This policy is useful for lowering the scheduling effort.

largest -- Among all the taskers that can execute a job, choose the tasker with the most amount of unused slots and unused RAM ((MB of unused RAM) + 16000*(Number of unused cores)). This policy tends to spread the jobs on idle machines. In most cases, this policy may not be the most effective.

smallest -- Among all the taskers that can execute a job, choose the tasker with the least amount of unused slots and unused RAM ((MB of unused RAM) + 16000*(Number of unused cores)). This policy tends to pack the jobs on machines that are already busy, thus keeping idle machines available if a large job is submitted.

smallram -- Among all the taskers that can execute a job, choose the tasker with the least amount of total RAM. This policy is useful to pack the jobs on the smaller machines first, which keeps large machines available if a large job is submitted.

At most one of these (Linux-only):

pack -- NUMA control: assign the job to a NUMA node with the least number of available resources that will fit the job. If none of the NUMA nodes have sufficient job slots and RAM, the job will be allowed to run on as many NUMA nodes as needed to satisfy its resource requirement.

spread -- NUMA control: assign the job to a NUMA node that has the largest number of available resources.

none -- NUMA control: allow Linux to place jobs. The Linux CPUs Allowed affinity list will be all the CPUs on the system (default).

Examples:

-jpp slowest
-jpp spread
-jpp smallest,pack

```
-jpp first,spread

Note: To place jobs on the same machines,
      use first or smallest.
-J <jobname>      -- Set the job name (same as -N <jobname>)
-keep             -- Keep job after completion, disabling auto-forget.
-keepfor <time>  -- Keep job after completion for specified time.
                  Disables auto-forget.
-limit <spec>    -- Add limit to jobs submitted with this option. <spec>
                  could be just a number, or a name followed by a
                  number. Throttles the running jobs of a user
                  submitted with the same -limit option to the
                  specified number.
-L <exitstatus>  -- Legal exit status list (default is 0). You can also
                  use commas to separate the valid statuses.
                  Examples:
                  -L 0,2,10          -L 0,200-208
-maxresched <N>  -- Maximum number of times the job can be rescheduled.
                  Must be >= 1 and <= 10. Implemented
                  via the MAX_RESCHEDULE property on the job.
-mpres RESLIST   -- Specification of the resources required by a
                  multiphase job. The RESLIST specifies the resource
                  lists for all phases of the job with % characters
                  delimiting the phases. The sublists of resources for
                  each phase are percent sign delimited.
                  Example: -mpres "RAM/200 CORES/2%RAM/20 CORES/3"
-mpres+ <rsrc>   -- Append one resource to the multiphase resource list.
                  This option must follow the "-mpres" or "-mpres<n>"
                  option, otherwise the resources specified in "-mpres+"
                  will be overwritten.

-mpres1 RESLIST
-mpres2 RESLIST
-mpres<n> RESLIST -- Specify resources for a stage <n> of a multiphase
                  job. The number <n> is in the range from 1 to 9.
                  Example: -mpres1 "RAM/200"
                           -mpres2 "CORES/4 RAM/10"
-N <jobname>     -- Same as -J <jobname>.
                  (compatible with FDL 'N' procedure)
-p <priorities>  -- Set priorities for scheduling and execution of job.
                  Format is <schedulingPriority>[.executionPriority]
                  Priority is either a number from 1 (low) to 15 (top)
                  or a symbolic value 'low' 'normal' 'high' 'top' or
                  any abbreviation thereof.
                  Examples: -p n      -p 4.high  -p high.low
-pre <SCRIPT>    -- Execute <SCRIPT> as precondition. The JOBID will be
-precmd <SCRIPT> appended to the arguments of the script. If the
                  script exits with non-zero status, the job is not
                  run. See examples in $VOVDIR/etc/pre.
-post <SCRIPT>   -- Execute <SCRIPT> as postcondition. The JOBID and
-postcmd <SCRIPT> EXITSTATUS will be appended to the arguments of the
                  script. The script is executed irrespective of the
                  success of the job. The exit status of the script
                  becomes the exit status of the job.
                  See examples in $VOVDIR/etc/post.
-preemptable <N> -- Set preemptable mode:
                  N=0    not preemptable
                  N=1    preemption allowed (default)
-profile         -- Activate job profiling to track and graph over time
                  the following: RAM usage, CPU usage, cumulative I/O,
                  and License usage. Without this option, only the
                  current usage of RAM, CPU, and Licenses are reported
                  in the web UI.
-r <r1> [r2..rN] -- Set requested resources of the job. Accepts
```

```
multiple resource arguments and may be repeated.
If -r is the last option, use '--' to separate
the last resource from the command line.
-r+ <resource> -- Append one resource to resource list.
                No termination necessary.
-r- <resource> -- Remove one resource from the resource list.
                It is an error to remove a resource that does not
                exist. No termination necessary.
-rf            Add Filer:<FILER_NAME> resource
                (computed from run dir)
-reconcilemem -- Monitor a job's actual memory usage and decrease
                consumed resources for the job if the consumed RAM for
                the job is less than the requested RAM. Optional value
                is a triplet of time specs:
                start[:interval[:end]]
                where:
                - start is the time after job starts that monitoring
                  begins.
                - interval is the time between checks. Default is
                  to check only once.
                - end is the time after job start that monitoring
                  ends. Default is end of job.
                For example, "-reconcilemem 1m:2m:8m" instructs
                Accelerator to start checking memory usage 1
                minute after the job has begun, do it every 2
                minutes and stop when the job has been running
                for 8 minutes. Note: if actual memory usage
                for a job exceeds requested RAM, the consumed
                RAM resource for the job is increased whether
                or not -reconcilemem is specified.
-rundir <dir>  -- Specify a different run directory (default ".")
                If the <dir> specification is quoted by single
                quotes, the directory is taken exactly as given,
                instead of being canonicalized. When using -rundir
                with the SNAPSHOT environment, the -ep argument
                must also be passed. Implies -D.
-set <setName> -- Assign the job(s) to the given set.
-sg <subgroup> -- Specify a subgroup for fairshare for the current user
-splitstderr  -- Write the stderr output of the interactive job to
                stderr. Default is to write the job's stderr output
                to stdout. Note that using this option will probably
                result in garbled terminal output due to interleaving
                of stdout and stderr outputs.
-tool <toolName> -- Specify a "toolName" different from the tail of the
                first command line argument. The argument must be
                less than 100 characters long and contain only
                alphanumeric chars.
-x | -xdur <xdur> -- Set the expected duration of the job.
-deadline <duration> -- Job is expected to be completed within the
                given duration.
                Set it to zero to disable it (the default).
-deadlineat <time> -- Job is expected to be completed before the
                given time.
                The time is parsed by the Tcl command
                [clock scan $time]
                Set it to zero to disable it (the default).

SUBMISSION OPTIONS:
-after <time>  -- Fire job after specified time.
-array <n>     -- Submit a jobarray of 'n' repeated commands
                Some fields may contain the strings @INDEX@, @JOBID@,
                and @ARRAYID@, which are substituted when the array
                is created.
```

```
These fields are: command, env, wd, toolname, jobname
The output files are also subject to the same
substitutions.
Three comma-separated formats for <n> are supported:
last
first,last
first,last,increment
-at <date>      -- Specify earliest date to fire job
                The date is parsed by the Tcl command
                [clock scan $date]
-atomic        -- Create job array using a single RPC between
                client and server.
-dp N          -- Run a Distributed Parallel (DP) job requiring N
                components.
-dpactive <n>  -- The n-th component is the one that becomes active
                (default 1).
-dpres RESLIST -- Specification of the resources required by a parallel
                job. Example: -dpres "RAM/200 CORES/2"
                See vovcreatepartialjobs for more info.
-dpres+ <rsrc> -- Append one resource to the distributed processing
                resource list.
                No termination necessary.
-dpres1 RESLIST
-dpres2 RESLIST
-dpres<n> RESLIST -- Specify resources for a component <n> of a DP job.
                The number <n> is in the range from 1 to <N>
                (option -dp)
                Example: -dpres1 "RAM/200"
                        -dpres2 "CORES/4 RAM/10"
-dpwait TIMESPEC -- The time the components wait to rendezvous
                (default 30s, minimum 3s). The wait is increased with
                each attempt. The maximum wait is controlled by the
                property DP_WAIT_MAX
-dpnocohortwait -- Partial jobs may exit without waiting for primary.
-dpinitialport N -- Specify starting port on which partial jobs should
                attempt to communicate.
-D            -- Do not check the validity of the directory.
-f <file>    -- Get a list of commands from a file, one per line.
                Jobs are created and then scheduled in blocks
                of 200 jobs (unless otherwise specified by -fb).
-fb <n>      -- Change the size of blocks of jobs scheduled with -f
                (default 200).
-fw <S>     -- Specify delay between blocks of jobs, in seconds.
                Value must be >= 0, default is 0. Use with -f.
-dribble     -- Short hand for -fb 1 -fw 0.1
-F          -- Force running of job even if it is already valid.
                This is useful only if you are also using option -l
                to set the name of the log file, otherwise this
                option has no effect.
-multiphase <N> -- Set multiphase mode:
                N=0    not multiphase (default)
                N=1    multiphase
                If a job will have more than 4 phases, then the
                "-maxresched N" option must also be specified, where
                N is the number of phases the job will have.
```

LOGFILE AND OTHER DEPENDENCIES OPTIONS:

```
-dep <Id|Name> -- Specify a dependency on the list of jobs.
-d <Id|Name>  -- The argument can be a list of job Ids or job names.
                In the case of job names, the dependency is looked
                for in the set of jobs belonging to the submitting
                user.
                The current job will not start until the
```

```
specified jobs have completed successfully.
May be repeated.
Performance note: dependencies on job names are much
slower than dependencies on job ids.
-depset <Name> -- Specify a dependency on all jobs in the named set at
the time of submission. If other jobs are added to
the set later, they will not be added to the
dependencies. May be repeated.
-forcelog -- Force the declared output log to be the output of
-force -- this job. If another job was declaring the same
output, it will become black (SLEEPING).
-forcedequeue -- Force the declared output log to be the output of
this job. If any job was declaring the same
output, upcone of all the jobs producing this
file will be stopped and dequeued, it will
become black (SLEEPING).
-i <in_file> -- Specify an input dependency.
-l <logfile> -- Specify name of logfile.
As with -rudir, if the <logfile> is quoted with
either " or ', then the name is taken literally
and not canonicalized.
Quoted or not, variable substitution on the file name
is performed for the following variables
@JOBID@ -> Id of job.
@ARRAYID@ -> Id of job array (if applicable).
@DATE@ -> ISO_TIMESTAMP
@UNIQUE@ -> %Y%m%d_%H%M%S.SUBMISSION_PID
@JOBCLASS@ -> job class (the alphanumeric part)
@JOBNAME@ -> job name (the alphanumeric part)
You may need to use -forcelog together with -l.
Timestamp in format '%Y%m%d_%H%M%S.SUBMISSION_PID'
will be added to the logfile name for array jobs
when '@UNIQUE@' is not present in the logfile name.
-n -- Use no wrapper (default: use 'vw').
-nolog -- Do not keep a log.
-o <out_file> -- Specify an output dependency.
-P <NAME=VALUE> -- Add the given property to the jobs (may be repeated).
-s -- Declare that the logfile is SHARED (see docs).
You rarely need this option. If misused, this option
causes extra buckets to be created in the scheduler.
Probably you need '-forcelog -F' instead.
-uniqueid -- Force NC to use a unique new
VovId for each job submission,
even when the same job is submitted multiple times.
-wrapper <W> -- Use specified wrapper '<W>' (default: use 'vw').

E-MAIL NOTIFICATION AND WAIT OPTIONS:
-m -- Send me mail upon job completion.
-M <mail rule> -- Send mail according to the given rule (see docs).
-w -- Wait for the job(s) to finish: do not show any log.
For the meaning of the exit status, check nc wait.
-wl -- Wait for the job(s) to finish: show the log of
the last job.
For the meaning of the exit status, check nc wait.

EXTRAS:
-noddb -- The job is not stored in the jobs log or in the
database.
-nopolicy -- For ADMIN only. Disable the policy layer.

EXAMPLES:
% nc run sleep 10
% nc run -autokill 30m sleep 1000000
```




```
% nc run -r SLOT/4 -xdur 500 -deadline 1h sleep 500
% nc run -r SLOT/4 -xdur 500 -deadlineat 10am sleep 500
% nc run -array 10 sleep 1 # submit 10 sleep jobs via
% nc run -array 10,200,10 sleep 1 # submit sleep jobs with index
% nc run -g /teams/chipA -sg session12 sleep 1
% nc run -G /teams/chipA.any sleep 1
% nc run -C longjobs sleep 10000
% nc run -C longjobs -r+ RAM/200 sleep 10000
% nc run -r unix -- sleep 10
% nc run -p high sleep 10
% nc run -e BASE -p h sleep 10
% nc run -e SNAPSHOT+SIM -p h sleep 10
% nc run -m sleep 10; # email when job finishes.
% nc run -M ":ERROR" sleep 10; # email only if
% nc run -dp 3 -dpres sun7,linux vovparallel clone sleep 10
% nc run -at 6pm sleep 10
% nc run -at "tomorrow 6pm" sleep 10
% nc run -after 10m sleep 10
% nc run -forcelog -F -l mylog.txt ./myjob
```

Default Output of nc run

The default output from `nc run` includes the following information:

- The [resource list](#) assigned to the job, which can be controlled with the option `-r`.
- The [environment](#) used for the job, which can be controlled with the option `-e`.
- The command line.
- The log file used to store both `stderr` and `stdout` of the command, which can be controlled with the option `-l`.
- The JobId assigned by Accelerator to this job. [JobIDs](#) are used as handles with many of the Accelerator commands.

Submit a Single Job

 **Note:** In the job examples provided, each job performs `sleep xx`, which is wait, do nothing, for the duration specified by the integer value `xx`.

When submitting a single job:

- Use an environment snapshot.
- The default resource list is the `vovarch` of the machine that submits the job.
- The name of the log file is automatically computed.

Run a Job, No Specifications

```
% nc run sleep 30
Resources= sun5
Env = SNAPSHOT(vnc_logs/env912488489.csh)
Command = sleep 30
Logfile = vnc_logs/20020704/150449.6528
JobId = 00002539
```

Run a Job, Environment Specified (-e)

```
% nc run -e BASE sleep 30
...
```

Run a Job, Environment and Resources Specified (-r)


```
% nc run -e BASE -r linux -- sleep 30
Resources= linux
Env = BASE
Command = sleep 30
Logfile = vnc_logs/20020704/150515.6528
JobId = 00002544
```

Run a Job, Environment and Resources Specified, Limited Verbosity (-v)

```
# Control verbosity: print the jobId only.
% nc run -v 1 -e BASE -r linux -- sleep 30
00002579
# Running a job, environment and resources specified, limited verbosity and wait for
job to finish (-w):
% nc run -w -v 0 -e BASE -r linux -- sleep 30
```

Submit Multiple Jobs

When a list of similar jobs is to be submitted, it is much more efficient to submit them all at once.

 **Note:** When submitting multiple jobs use the same environment and the same resources, and the same priority level is scheduled for each job. Each job has its unique identification.

1. Prepare a file with one command on each line. Empty lines are ignored and lines that begin with # are considered comments.

```
# Example of file used to submit multiple jobs at once.
sleep 10
sleep 11
sleep 12
sleep 13
```

2. Use the option -f to specify the command file, as in the following example:

```
% nc run -r unix -e BASE -f commandFile
```


All jobs submitted with this method share the same environment, the same resources, and are scheduled at the same priority level. Each job has its own ID.

Submit Jobs from the Browser

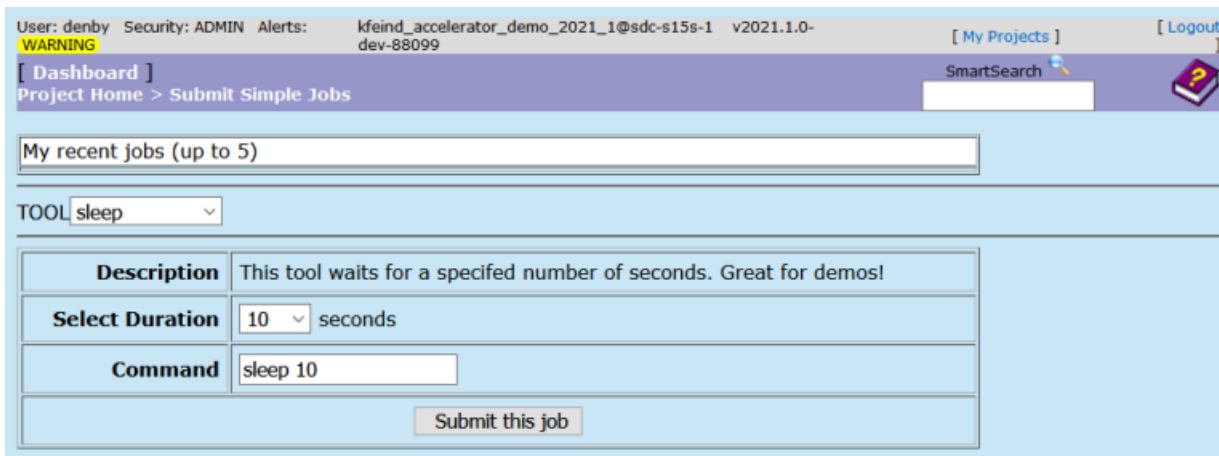
This section shows an example of using a web browser to submit jobs.

For information about using the command line interface to submit jobs, refer to *Submit Jobs with CLI Commands* in the Altair Accelerator User Guide..

An example script is located in `$VOVDIR/etc/cgi/submit.cgi`, which can be modified as needed. The script is invoked with the URL `http://HOST:PORT/cgi/submit.cgi`, which is usually `http://nc-host:6271/cgi/submit.cgi`. (6271 is the default port; however, the port that is actually used can be different.)

 **Note:** Placing the file in `$VOVDIR/etc/cgi` will make it available to all servers running on that Altair Accelerator software hierarchy. To restrict access to Accelerator, create a `cgi` subdirectory under your Accelerator directory `vnc.swd`, and place the `submit.cgi` script there. Remember to make the script executable. `submit.cgi` is setup for running very basic scripts. This script is intended as an introduction to Accelerator.

The following is the output of the example script:



The screenshot shows a web browser interface for submitting jobs. At the top, there is a navigation bar with "User: denby", "Security: ADMIN", "Alerts: kfeind_accelerator_demo_2021_1@sdcs15s-1 v2021.1.0-dev-88099", "[My Projects]", and "[Logout]". Below this is a "WARNING" banner. The main content area has a "Dashboard" link and "Project Home > Submit Simple Jobs". A "SmartSearch" box is visible. The main section is titled "My recent jobs (up to 5)". Below this is a "TOOL" dropdown menu set to "sleep". A table with three rows is shown: "Description" (This tool waits for a specified number of seconds. Great for demos!), "Select Duration" (10 seconds), and "Command" (sleep 10). A "Submit this job" button is at the bottom of the table.

Figure 1:

View Job Progress and Results

Submitted Job Information Display

By default, `nc run` provides information when a job is submitted.

The following example shows information that is output with a simple command. The amount of information displayed is determined by the verbose level. In the following example, verbose is at the default level of 4.

 **Note:** The environment is set with a snapshot.

```
% nc run sleep 10
```

```
Fairshare= /time/users.andrea
Resources= macosx CPUS/1 RAM/500
Env = SNAPSHOT(vnclogs/snp/joe/macosx/env27227.env)
Command = vw sleep 10
Logfile = vnc_logs/20130220/104930.33137
JobId    = 024609542
```

- **FairShare:** the FairShare ranking of this job.
- **Resources:** the resources used to run this job: the machine, number of CPUs, amount of RAM, and so on.
- **Env:** the environment in which the job was submitted.
- **Command:** the command that was used to execute this job.
- **Logfile:** the name of the logfile.
- **JobID:** the unique identifier of this job.

The amount of information can be changed by setting the verbose level by using the `-v` option, such as

```
nc run sleep 10 -v 2
```

Verbose Level	Effect
0	Silent: no output is generated
1	Only the job ID is displayed
4	Default level
9	Very verbose

Get Job Information

The `nc info` command gives information about a job.

When executed without an option, the subcommand `info` displays the start time, completion time, duration, resources, priority, and exit status.

```
vnc: Usage Message
NC INFO:
  Get information about a specific job or list of jobs.
USAGE:
  % nc info [options] <jobId> ...
OPTIONS:
  -h                -- Show this message.
  -v                -- Increase verbosity.
  -l                -- Show the log file (actually, it shows all outputs).
  -ioprofile        -- Show job I/O profiling summary statistics for jobs
                    that have been completed. The job must have been
                    submitted with the -ioprofile option.
```

```
-e          -- Show the environment name, or contents if a snapshot.
-c          -- Show tasker compatibility table
             (which tasker can run a job).
-sc         -- Show tasker compatibility in normal output.
-tc         -- Show tasker compatibility in normal output.
-dep        -- Show job dependencies.
-J <jobname> -- Show the jobs with given name.
             (DO NOT USE IN SCRIPTS!)
```

NOTE: 'nc info -J JOBNAME' is way more expensive than 'nc info JOBID'
We strongly recommend you never use nc info -J ... in scripts.
Reason: job names are not unique and are not hashed. Job Ids are unique.

EXAMPLES:

```
% nc info 00123456
    -- Show info about specific job.
% nc info !
    -- Show info about most recent job in current dir.
% nc info -l 12345
    -- Show log file(s) of job.
% nc info -J MyJob
    -- Show info about all jobs called "MyJob".
% nc info -tc 0012345
    -- If job is Scheduled, also show the summary
       of tasker compatibility.
```

Get Detailed Information about a Job

The command nc displays information about a job.

Get Detailed Information About a Job

The command nc getfield also gives information about a job, but in an undecorated form that is in scripts.

nc: Usage Message

NC GETFIELD:

Get one or all fields of one or more Accelerator jobs. Specify the jobID
or use '!' for the most recent job in the current working directory.

If the -J jobName option is given, only the first match
is reported. If there is no match, an error is reported.

OPTIONS:

```
-f field    -- Specify field when giving multiple jobIDs.
-h          -- Help usage message. You can also get the usage message by
             specifying no option at all.
-J JOBNAME  -- Find first job with given JOBNAME. The search is restricted
             to the jobs that belong to the current user. This is
             significantly more expensive than using jobIDs. Use
             sparingly.
-s          -- Same as -showid.
-sep STRING -- Use STRING as separator (default is a single space).
-showid     -- Show jobId.
-tab        -- Use a TAB character as separator.
-v          -- Increase verbosity.
```

EXAMPLES:

```
% nc getfield -h
% nc getfield 01234455
% nc getfield 00123445 jobclass
% nc getfield ! status
% nc getfield -J JOBNAME
% nc getfield 01234455 0123458 -f jobclass
% nc getfield -s 01234455 0123458 -f jobclass
```

Examples:

```
% nc getfield 00012345 jobclass
normal
% nc getfield 00012345 cputime
7.125
% nc getfield 00012345
... get list of all known fields (more than 100 of them)...
```


Graphical View of Job Progress and Results

Job Status

In Accelerator, each job goes through a number of states until completion.

The states are described in the following table:

Status	Color	Description
Idle	BlueViolet	If the node is a job, either it has not been run successfully yet or it needs to be run again, because one of its inputs has been modified since the last time the job was executed. If the node is a file, it is the output of a job that is not Idle.
Queued	Light blue	The job is scheduled to be run. It may be already queued or it will go in the queue as soon as all its inputs are ready.
Running	Orange	The job is currently being retraced; it has been dispatched to one of the taskers. All the outputs of such a job are either RETRACING or RUNNING.
Done	Green	If the node is a job, it has run successfully. If the node is a file, it is up-to-date with respect to all other files and jobs on which it depends.
Failed	Red	The job ran and failed.
Transfer	Cream	The job is being transferred to another cluster and it is not yet running.

Status	Color	Description
Suspended	Pink	The job was running (or retracing) and one of the processes belonging to the job is currently suspended.
Sleeping	Black	Either the job caused an output conflict upon submission (bad dependencies) or the job was not reclaimed by any tasker upon crash recovery.
Withdrawn	Gray	A job has been withdrawn after dispatching, such as by the preemption daemon.  Note: This status occurs rarely and tends to be hard to observe.

The normal sequence for a successful job is **Idle > Queued > Running > Done**

The normal sequence for a failing job is **Idle > Queued > Running > Failed**

Set Status

The status of a set is computed from the status of the nodes it contains.

The status of a set is computed from the status of the nodes it contains. If the set is empty, it's status is `EMPTY`. Otherwise, the status is determined by the *dominant* status according to the following ranking:

1. `RUNNING`
2. `RETRACING`
3. `FAILED`
4. `INVALID`
5. `UNKNOWN`
6. `MISSING`
7. `DELETED`
8. `SLEEPING`

If at least one node is `RUNNING`, then the set itself is `RUNNING`. Otherwise, if at least one node is `RETRACING`, then the whole set is `RETRACING` and so on.

In particular, if the set status is `VALID` (that is, green), then the set has no `INVALID`, `FAILED`, `RETRACING` or `RUNNING` nodes.

The utility `vovset` can be used to create, modify, recompute, destroy sets.

Manage Jobs

Accelerator provides several commands of job controls, including wait, stop, cleaning and debugging.

Wait for Jobs

To wait for one or more jobs to complete, use the `nc wait` command.

nc wait

Wait for a specific job or list of jobs to complete, fail, or be forgotten.

```
nc: Usage Message

NC WAIT:
    Wait for a specific job or list of jobs to complete,
    fail, or be forgotten
USAGE:
    % nc wait [OPTIONS] <jobId> ...
OPTIONS:
    -h                -- Print this message.
    -q                -- Quiet wait.
    -v                -- Verbose (may be repeated up to 3 times)

    -dir <directory> -- Wait for all jobs in the given directory.
    -subdirs <directory> -- Wait for all jobs in the given directory and
                        subdirectories.
    -select <rule>    -- Wait for all jobs in set defined by 'rule'.
                        The predicate 'isjob' is automatically added
                        to the rule.
    -set <setName>   -- Wait for all jobs in a set.
    -start           -- Wait for the specified jobs to start.
    -p              -- Use polling method (automatic delay)
    -poll <MS>      -- Use polling method with a delay of <MS>
                        milliseconds.
                        MS values are between 2000 and 60000.
    -k              -- Jobs are killed if you interrupt wait
                        by Ctrl-C.
    -l              -- Show log file of last job while waiting.
                        The log is shown either as it is created
                        if the job has been submitted with option -wl
                        or all at once upon job completion.
    -maxwait <timespec> -- Specify a maximum wait time.
    -timeout <timespec> -- Same as -maxwait
    -jobinfo        -- Show info about jobs being executed.
    -callback <cmd> -- Execute 'cmd JOBID JOBSTATUS' for every job
                        that completes. Output to stdout.
                        Errors ignored.
    -file <file>    -- For experts. Source <file>, mostly to define
                        overrides for procedure VncWaitCallback
                        { cmd jobId status }
```

RETURN:


```
0      -- All jobs are done, or started if -start is specified.
1      -- Some jobs are still invalid.
2      -- Some jobs are failed.
3      -- Some jobs have unexpected status.
4      -- Some jobs have been lost.
5      -- Waited for long enough (see -maxwait option)
```

EXAMPLES:

```
% nc wait 22345 22356
% nc wait -dir .
% nc wait -subdirs .
% nc wait -select "command~spice"
% nc wait -set "myset"
% nc wait -callback ./myScript -set myset
% nc wait -jobinfo -set myset
% nc wait -p -set myset
% nc wait -poll 5000 -set myset
% nc wait -maxwait 1m -set myset
```

Stop Jobs

A job can be stopped when it is running or queued. Stopping a job does not "forget it" from the vovserver database. A job can only be stopped by the owner or the Accelerator administrator.

nc stop

nc: Usage Message

NC STOP:

Stop jobs.

1. If the jobs are running, they are killed (unless you use option `-dequeueonly`).
2. If the jobs are scheduled in the queue, they are removed from the queue.

In either case, the jobs remain in the system. To remove them from the system, use the "forget" command. Jobs in the system can be rerun with the "rerun" command.

When stopping a single job, the procedure checks for the properties `NC_STOP_SIGNALS` and `NC_STOP_SIG_DELAY` attached to the job to be stopped.

The list of signals is also controlled by the environment variables `VOV_STOP_SIGNALS` and `NC_STOP_SIGNALS`. If both `NC_STOP_SIGNALS` and `VOV_STOP_SIGNALS` are present in the environment, the value of `VOV_STOP_SIGNALS` will be used. Their functionality is otherwise identical.

The default list of signals is `TERM,HUP,INT,KILL` and can be customized with the variable `defaultStopSignalCascade` in `policy.tcl`.

USAGE:

```
% nc stop [OPTIONS] <jobId> ...
```

OPTIONS:

```
-after <s>      -- Start sending signals after specified seconds.
                  This is an initial delay, between 0 and 20s.
```

```
-allusers          -- Stop all jobs (only ADMIN can do it).
-d                -- Same as -dequeueonly.
-delay <s>        -- Minimum delay between signals (in seconds),
                  between 0 and 20s. Default is 3.
                  This can also be set with the property
                  NC_STOP_SIG_DELAY, or with the environment
                  variables NC_STOP_SIG_DELAY or
                  VOV_STOP_SIGNAL_DELAY. If both NC_STOP_SIG_DELAY
                  and VOV_STOP_SIGNAL_DELAY are present in the
                  environment, the value of NC_STOP_SIGNAL_DELAY
                  will be used.
                  Priority: 1. Option -delay
                           2. job property NC_STOP_SIG_DELAY
                           3. env variable VOV_STOP_SIGNAL_DELAY,
                              NC_STOP_SIG_DELAY
                           4. default

-dequeueonly      -- Just remove jobs from the queue.
                  All currently running jobs are not affected.
                  Can be abbreviated to -d.

-dir <directory> -- Stop all jobs in the given directory.
-exclude <PROCLIST> -- List of processes to exclude from receiving the
                  signal.
-h               -- This message
-include <PROCLIST> -- List of processes to receive the signal.
-J <jobname>     -- Stop all my jobs with given jobname.
-mine           -- Stop all my jobs.
-set <setname>   -- Stop all my jobs in the given set.
-sig <SIGLIST>  -- Same as -signals.
-signals <SIGLIST> -- Comma separated list of signals to send to the jobs
                  (default is the sequence TERM,HUP,INT,KILL )
                  This can be also set with property NC_STOP_SIGNALS
                  or with the environment variables NC_STOP_SIGNALS
                  or VOV_STOP_SIGNALS.
                  Priority: 1. Option -signals
                           2. job property NC_STOP_SIGNALS
                           3. env variable VOV_STOP_SIGNALS,
                              NC_STOP_SIGNALS
                           4. default (can be configured as
                              defaultStopSignalCascade
                              in policy.tcl)
                  See also: vovshow -env VOV_STOP_SIGNALS
                           vovshow -env NC_STOP_SIGNALS

-skiptop <0|1>   -- Whether to kill the top process.
                  This is normally the job wrapper (e.g. vw, vwi).
                  Default is 0.

-why <reason>    -- Give a reason for the stop.
                  This is stored on the WHYSTATUS
                  field of the stopped jobs.
```

EXAMPLES:

```
% nc stop 00123456
% nc stop -d -mine
% nc stop -after 3 -mine
% nc stop -set Class:hsim
% nc stop -mine -why "Jobs no longer needed"
% nc stop -sig "TERM,KILL" -delay 4 0012345
% env VOV_STOP_SIGNALS=TERM,INT,KILL nc stop 0012345
```

SEE ALSO:

```
% vovshow -env VOV_STOP_SIGNALS
% vovshow -env NC_STOP_SIGNALS
% vovshow -env VOV_STOP_SIGNAL_DELAY
% vovshow -env NC_STOP_SIG_DELAY
```

Override Signals to Stop a Job

A job can be stopped by overriding the sequence of signals that are sent for the job. To do so, set the properties NC_STOP_SIGNALS and NC_STOP_SIG_DELAY.

Automatic Stopping Based on Elapsed Time

If a job is submitted with the -autokill option, it will be stopped after the specified amount of time has elapsed. The check to stop the job is performed by the tasker itself at an interval of about one minute, which can be controlled with the -U option of vovtasker).

For example:

```
% nc run -autokill 30m sleep 1000000
```

Automatic Stopping Based on CPU Time

To stop a job that exceeds a specific duration of CPU time, set the variable VOV_LIMIT_cputime. A job that exceeds the limit will be killed by UNIX and will have status "Failed".

For example:

```
% nc run -e "BASE+D(VOV_LIMIT_cputime=10)" vovmemtime 10 100 0
```

Forget Jobs

Under normal operation, jobs are automatically forgotten from the server database as follows:

1. Completed jobs are forgotten after one hour.
2. Failed and idle jobs are forgotten after two days.
3. Queued and running jobs are never forgotten.

The nc forget command immediately deletes the specified job from the server database. If a job is running and the -forcerunning flag is used, the job is stopped before it is forgotten.

nc forget

Forget jobs from the trace.

```
vnc: Usage Message  
  
NC FORGET:  
Forget jobs from the trace.  
If the jobs are running they are first stopped (if you use -forcerunning)  
If the jobs are queued, they are removed from the queue.
```

```
USAGE:
  % nc forget [OPTIONS] <jobId> ...

OPTIONS:
  -normal          -- Forget all my jobs older than 1 day.
  -n              -- Shortcut for -normal.
  -age <age>      -- Forget all my jobs older than the
                   specified age (except running jobs).
  -J <jobname>    -- Forget all my jobs with given name.
  -set <setname>  -- Forget all jobs in given set.
  -mine           -- Forget all my jobs (regardless of age).
  -allusers       -- Forget jobs belonging to other users too.
                   Need to be ADMIN.
  -dir <dirname>  -- Forget all jobs in the given directory.
  -subdirs <dirname> -- Forget all jobs in the given directory and
                   all subdirectories.
  -selrule <rule> -- Selection rule for jobs to forget.
  -forcerunning  -- Force deletion of running jobs.
  -h             -- This message.
  -v             -- Increase verbosity.
  -quiet         -- Quiet forget. Ignore errors.
  -system        -- Include system jobs (implied for explicit jobIds)

EXAMPLES:
  % nc forget -n
  % nc forget -age 1h
  % nc forget -mine -dir .
  % nc forget -allusers -dir .
  % nc forget -set MyExperiment
  % nc forget -set MyExperiment -forcerunning
```

Autoforget Jobs

The autoforget flag sets up a job to automatically be forgotten by the system after a certain time, (not including suspension time) if and only if the job is done, failed, or idle. Jobs that are scheduled, running, suspended or transfer are never autoforgotten.

Global auto-forget Parameters

There are three different auto-forget parameters:

- *autoForgetValid*
- *autoForgetFailed*
- *autoForgetOthers*

In Accelerator, the autoforget flag is set by default, which can be unset by using the option `-keep` in `nc run`. In Flow Design Language (FDL), the variable `make(autoForget)` controls the flag.

- The autoforget flag on the job is true
- The job is done, failed, or idle.

Jobclass Specific auto-forget

- The job belongs to a jobclass with the `AUTOFORGET` property set to a positive value.
- The job is done, failed, or idle.

 **Note:** The autoforget flag on the job is irrelevant.

If a jobclass has a specific auto-forget property, then the jobs in that jobclass will be forgotten after that specified time.

For example, to set the autoforget property on a jobclass called *abc*, use the `vtk_jobclass_set_autoforget` API:

```
% nc cmd vovsh -x "vtk_jobclass_set_autoforget abc 2m"
```

To disable this functionality for a jobclass, set the value of autoforget to a non-positive value, such as:

```
% nc cmd vovsh -x "vtk_jobclass_set_autoforget abc 0"
```

Auto-forget Log Files

If the parameter `autoForgetRemoveLogs` is true and the parameter `disablefileaccess` is false, the vovserver tries to delete the log file of the jobs that are being auto-forgotten. The success of the deletion depends on the file permissions.

 **Note:** Accessing files makes the vovserver vulnerable to NFS problems.

Auto-forget Examples

For this example, the default autoforget policy is to forget jobs after 1h. Other jobs in the jobclass "Regression" should be retained for 10days. Submit the jobs with the `-keep` option (no autoforget flag) and then set the `AUTOFORGET` property in the set `Class:Regression` to 864000.

```
### Done by an ADMIN
% nc cmd vovsh -x 'vtk_jobclass_set_autoforget Regression 10d'

% nc run -C Regression -keep ./my_test
```

Conversely, if the retention policy keeps the jobs for a long time (such as 3 days), some jobs in the jobclass "Quick" may be set to be forgotten more promptly, (such as after 5m) of completion. In this case, set the `AUTOFORGET` property in the jobclass set as follows:

```
### Done by an ADMIN
% nc cmd vovsh -x 'vtk_jobclass_set_autoforget Quick 5m'
```

Rerun Jobs

The `nc rerun` command initiates scheduling and executing jobs that are already in the server database.

By default, only the jobs that are `idle` or `queued` are affected by this command. To force rerunning jobs that are either complete or failed, use the option `-F`.

Examples of rerunning jobs:

```
% nc rerun 2345 2355
nc: message: Scheduled jobs: 1 Total estimated time: 1m13s
nc: message: Scheduled jobs: 1 Total estimated time: 58s
% nc rerun -p 8 2345
```

```
nc: message: Scheduled jobs: 1 Total estimated time: 1m13s
% nc rerun -d .
nc: message: Scheduled jobs: 44      Total estimated time: 20m23s
```

nc rerun

Rerun jobs already in the system.

```
vnc: Usage Message

NC RERUN:
    Rerun jobs already in the system
USAGE:
    % nc rerun [OPTIONS] jobId ...
OPTIONS:
    -h                -- This message.
    -F                -- Force to rerun all specified jobs.
    -f                -- Same as -F (backwards compatibility).
    -force            -- Same as -F.
    -first            -- Put job at top of its bucket.
    -J <jobname>      -- Rerun all jobs with the given jobname.
    -set <setname>    -- Rerun all jobs in the given set.
    -dir <directory> -- Rerun all jobs in the given directory.
    -p <priority>    -- Set scheduling priority.
    -mine            -- Rerun all my jobs.
    -r <aux_resources> -- Additional resources to be used.
    -v                -- Increase verbosity level.

OBSOLETE OPTIONS:
    -all              -- Rerun all jobs.
```

Automatic Rerunning of Failed Jobs

If a job fails quickly, it is possible that a faulty tasker machine may be to blame. Under a faulty machine scenario, many jobs can quickly run and fail, potentially emptying the job queue, requiring those jobs to be manually rescheduled.

The main method of protecting against this scenario is [Black Hole Detection](#). If for any reason this method of protection does not provide adequate coverage, a secondary method of protection is provided: auto-rescheduling.

This feature instructs the scheduler to rerun a job that fails within a configurable time threshold on a different host or tasker (depending on parameter *autoRescheduleOnNewHost*) than the one on which it previously failed.



Note: This feature may cause unwanted results in cases where job durations can be short and under the configured threshold. Because of this, the feature is disabled by default.

To enable auto-rescheduling, set the threshold to a non-zero value. The threshold is controlled by setting `config(autoRescheduleThreshold)` in the `policy.tcl` file. The value is a [timespec](#): 3s is 3 seconds. The typical value for this parameter is 2 to 4 seconds.

The default behavior is to rerun on a different host, but the scheduler can be configured to target a different tasker instead, via the `config(autoRescheduleOnNewHost)` setting. The default value is 1, which avoids the entire host. Set the value to 0 to avoid the tasker. Normally, only one tasker runs on a given host, but in some scenarios, multiple taskers may be present on the same host. In this case, the job may be routed back to the same physical machine as before. If the problem that caused the failure was temporary, the job may succeed. If the problem still exists, the job will likely fail again and be auto-rescheduled to another tasker in the pool.

Before a job is automatically resubmitted, the negated name of the host or tasker where the job previously failed is appended to the job's requested resources. The job will not run again on a host or tasker where it has failed.

For example, if the job requests the resources `linux hsim` and fails on a host named `broken-host`, it will be resubmitted with the resources

```
linux hsim !broken-host
```

If the job fails again quickly on a host named `jupiter`, the resources will become

```
linux hsim !broken-host !jupiter
```

Automatic resubmission is allowed a limited number of times between 0 and 10, which is controlled by the following parameters in the order shown:

1. The property `MAX_RESCCHEDULE` attached to the job (if any);
2. The property `MAX_RESCCHEDULE` attached to the jobclass set. This property can be set with `vtk_jobclass_set_max_reschedule`;
3. The parameter `autoRescheduleCount` (default value 4), can be set in `policy.tcl`. An example follows:

```
# Fragment of policy.tcl
set config(autoRescheduleCount)      4
set config(autoRescheduleThreshold)  2s
set config(autoRescheduleOnNewHost)  1
```

Job Management

Job Arrays

Using an array provides the option of submitting multiple jobs in a specific order.

Each job in an array is assigned its own job ID and is treated as an individual job. The syntax is: `nc run -array <n>`

To submit an array, use option `-array N` in `nc run`. The value of `N` is between 1 and the value specified by the `maxJobArray` configuration parameter. `maxJobArray` is normally set to 1000.

For example:

```
% nc run -array 100 sleep 10
```

The job specification may contain the symbolic element `@INDEX@` in either the command line, the environment, or the directory specification. The `@INDEX@` element is substituted when the job array is created. Use `@INDEX@` in the command line of the job array or in its environment.

Examples


```
% nc run -array 100 -e "BASE" sleep @INDEX@  
nc run -array 100 -e "BASE+D(MYINDEX=@INDEX@)" sleep 10
```

VOV_JOBINDEX

When you submit a job array, such as:

```
nc run -array 5 myJob.sh
```

The `VOV_JOBINDEX` environment variable will be set in the execution environment of each job in the array. In the above example, the first job created will have a `VOV_JOBINDEX` value of 1, the second job will have 2, and so on, with the last job having a value of 5.

 **Note:** This variable is for consumption only and is not intended to be set by the user at any time.

Jobname Attribute

The jobname is an attribute of a job.

The default value is the null string "".

- The jobname does not need to be unique for each job.
- The jobname, if defined, is used in the GUI as the primary label for the jobs; otherwise, the tool name is used.
- The field associated with the jobname is `@JOBNAME@`.

To set the jobname in the Flow Design Language (FDL), use the procedure N. For example:

```
N "Clock Routing"; # A short name for the job.  
J vw /some/long/path/to/an/executable/script/script.pl arg1 arg2
```

Jobname in Accelerator

In Accelerator, the jobname can be set with the option -N. For example:

```
% nc run -N this_is_my_job sleep 100
```

For all products, strict job name checking has been enabled and invalid job name characters will cause an error. For Accelerator and Accelerator Plus, this can be overridden by putting the following in `$VOVDIR/local/vnccrun.config.tcl`:

```
set ::jobname_lexicon legacy or set ::jobname_lexicon replace.
```

Legacy will use the more lax job naming rules from earlier releases.

Replace will identify invalid characters in the job name, replace them with "_", and issue a warning to the console.

Jobclasses

A jobclass allows multiple job parameters to be set in a single object that can be requested at submission time.

For example, there may be a job that requires 3 different licenses, 4GB of RAM, and 4 cores. Instead of requesting all 3 licenses, a jobclass can be created that is called with the -C submission option to the `nc run` command. Jobclasses are often used to emulate queues that are found in other batch processing systems.

 **Note:** A jobclass can only be created by an Accelerator administrator.

Find the Available Jobclasses

To list the available classes from the command line, use the `jobclass` subcommand of the `nc` command.

```
% nc jobclass [OPTIONS]
```

The `jobclass` subcommand accepts the repeatable option -l. The first option includes the description, and the second option shows the values to which `VOV_JOB_DESC` slots will be set.

In addition, Accelerator provides the Jobclass page. This page shows a table of the job classes, with links to the definitions of each class, and to the sets containing the jobs in that class. It also shows the pass/fail status as a bar graph.

Submit Jobs Using Jobclasses

To submit a job in a given class, use the option `-C` of `nc run`.

```
% nc run -C short sleep 10
```

Jobs in a class are automatically added to a set named after the class, for example "Class:interactive".

The options to `nc run` are parsed sequentially, so it is possible to do a command line override of the parameters set in the job class. For example, the following commands behave differently:

```
% nc run -C verilog -e DEFAULT -- run_sim chip  
% nc run -e DEFAULT -C verilog -- run_sim chip
```

In the first invocation, the option `-e` overrides the specifications for the environment to be used for the job. In the second invocation, the environment is determined by the definition of the *verilog* jobclass.

Pre-Command and Post-Command Job Conditions

When a job is being submitted, a pre-condition and/or a post-condition can be specified.

- **pre-condition:** a script that is executed before the job is executed.
- **post-condition:** a script that is executed after the job has completed. The post-condition is typically used to perform cleanup, such as deleting temporary files in `/usr/tmp`.

Example scripts are available in the following directories: `$VOVDIR/etc/pre` and `$VOVDIR/etc/post`.

Pre-condition

A pre-condition is executed before the job is run. It is invoked with a single argument: the ID of the job. A pre-condition is executed with the same credentials as the job (userid, os-groupid) and is in the same directory of the job.

- If the precondition script fails by exiting with a status different from 0 (zero), the job will not be run and the exit status of the job will be the exit status of the pre-condition script.
- If the exit status of the pre-condition script is within the range 201-215, the automatic rescheduling condition will occur and the job will be rescheduled on a different host or on a different tasker.

Post-condition

The post-condition script is invoked with two arguments: the ID of the job and the exit status of the job. The post-condition is executed with the same credentials as the job (userid, os-groupid) and in the same directory of the job.

- When the post-condition script is invoked, the job is still running.
- The post-condition is executed after the job, even if the job fails, but it is not executed if the pre-condition fails.
- The exit status of the post condition overrides the exit status of the job. It needs to explicitly return the exit status of the job when that is the requested behavior (see the example scripts).

Submit Jobs with Conditions

Use the options `-pre` and `-post` with `nc run` to specify the pre- and post- conditions.

```
% nc run -pre $VOVDIR/etc/pre/pre_check.sh sleep 10  
% nc run -post $VOVDIR/etc/post/post_cleanup.sh sleep 10
```

Log Files

The standard output from the pre- and post-commands is saved in log files. The location of the log files is determined by the value of the environment variable `NC_LOGDIR`. If `NC_LOGDIR` is not set, the files are stored in the directory `./vnc_logs`, relative to the current launch directory.

In the following example, `NC_LOGDIR` is not set, and the run directory is `~/testrundir`:

```
[goetz@goetz1 ~/testrundir]$ pwd
/home/goetz/testrundir
[goetz@goetz1 ~/testrundir]$ ls
vnc_logs
[goetz@goetz1 ~/testrundir]$ ls -a vnc_logs/
. .. 20210726 .precmd.000083865.log .precmd.000083885.log snapshots
```

The log files are created with zero size if the pre- and post-commands redirect all the output of the files. At the end of the job, if these files are zero length, they are automatically deleted to reduce disk space overhead.


The log files are named according to the following rules:

```
.precmd.$jobID.log
.postcmd.$jobID.log
```

The pre- and post-command log files can optionally be located in the same directory as the job logfile. For example:

```
nc run -pre "myprecommand > @JOBLOGDIR@/@JOBID@_pre.out" -l path/to/an/existing/
directory/mycommand.out -- mycommand
nc run -post "mypostcommand > @JOBLOGDIR@/@JOBID@_post.out" -l path/to/an/existing/
directory/mycommand.out -- mycommand
```

This would result in the respective pre- and post-command logfiles being written to the directory `path/to/an/existing/directory`.

 **Note:** When using the `nc run` command after forgetting jobs that have pre- and/or post-commands, it does not automatically remove the pre- and post-command `.log` files. If these files are not zero length, they must be removed manually.

Schedule Job Submission

Queue Selection

A queue is a cluster, a group or farm of machines. Accelerator supports multiple queues that are managed by the same vovserver. Queue names begin with the letters *vnc*.

Start a Queue

The command to start a queue is `ncmgr start [queue-name]`. Entering a *queue name* is optional. By default, if no name is entered, the queue will be named *vnc*.

ncmgr start

```
vncmgr: Usage Message

USAGE:
    % ncmgr start [options]

OPTIONS:
    -h                This help.
    -force            Do not ask confirmation.
    -block            Do not return to the shell or command prompt
                    after starting. This is only useful, and
                    required, when starting Accelerator as a
                    Windows service.
    -port <port|mode> Specify port number, port number list (colon
                    separated) or port mode. Modes are:
                    automatic - hash queue name into port number,
                    do not start if port is
                    unavailable. The default queue
                    name 'vnc' hashes to port
                    6271.
                    any - hash queue name into port number, try
                    additional ports in increments of 1
                    until an available one is found. The
                    default queue name 'vnc' hashes to
                    beginning port 11437.
    -webport <port|mode> Default: any
                    Specify a dedicated web interface port for
                    HTTP and HTTPS protocols. This port must be
                    configured to enable REST API v3 interface,
                    to enable the dashboard web UI page,
                    and to enable SSL. A value of 0 directs
                    VovServer not to open a web interface port.
                    Specify port number, port number list (colon
                    separated) or port mode. Modes are:
                    automatic - hash queue name into port number,
                    do not start if port is
                    unavailable. The default queue
```

```
        name 'vnc' hashes to web port
        6271.
    any - hash queue name into port number, try
        additional ports in increments of 1
        until an available one is found. The
        default queue name 'vnc' hashes to
        beginning web port 9695.
    Default: Any
    -webprovider <provider> Specify the provider for
        HTTP and HTTPS protocols.
        This must be either "internal" or "nginx".
        Default: "internal"
    -roport <port|mode> Specify read-only guest access web interface
        port. A value of 0 disables this interface,
        requiring all web interface users to log in.
        Default: 0
    -q, -queue <name> Name for queue (default is $NC_QUEUE if set,
        and otherwise vnc).

    -dir <dir> Directory of the server
        (default $VOVDIR/../../vnc).
    -dbhost <host> Host for database.
    -dbroot <path> Path on database host for database files.
    -dbport <port> Port of the database to listen for
        connections.
    -v Increase verbosity.

EXAMPLES:
% ncmgr start -port 6271
% ncmgr start -port 6271:6272:6273:any -force
% ncmgr start -q bigqueue -dir /remote/queues
```

Select a Queue Name

The default name of the Accelerator queue is *vnc*. Unless otherwise specified, all Accelerator commands act on the default queue.

To request a different queue one command at a time, by use the *-q* option with the *nc* command.

Example:

```
% nc [-q queue -name]run[run -options]command
```

Change the Default Queue with *nc_queue*

A different queue can be specified with using the *-q* option with the *nc* command.

To change the default Accelerator queue from *vnc*, set the environment variable *NC_QUEUE* to the queue name.

Example:

```
% setenv NC_QUEUE vnc_regr
```

Scheduled Jobs

Jobs that can not be dispatched immediately due to resource shortage, such as CPUs or software licenses, are put on the job queue.

Jobs are scheduled using the following rules:

- Scheduling is first determined by the **FairShare** mechanism. All active FairShare groups, all groups with queued jobs, are ranked based on their distance from the target share of computing resources and the current number of running jobs.

The FairShare group that is farthest behind the target has rank 0 (zero) and is selected first for scheduling. If none of the jobs from the FairShare group with rank 0 can be dispatched, Accelerator looks at the jobs for the FairShare group for rank +1 and so on.

- For a given FairShare group, jobs with higher **Priority** are scheduled ahead of lower priority jobs.
- For a given FairShare group of a given priority, jobs are scheduled on a first-come first-serve basis.

To check the status of the jobs in the queue, use the command `nc summary` or check `/jobqueue?action=buckets`. This page gives a report on all the classes of queued jobs (known as *buckets*):

- The characteristics of the bucket: user, group, priority, and tool.
- The number of jobs in the bucket and the age of the bucket: how long ago a job from that bucket was successfully dispatched.
- The resources the jobs are waiting for.

Understand Why Jobs are Queued

In addition to the overall information about the job queue and its buckets, you can also query individual jobs or sets, using the CLI, GUI, or browser.

- From the command line:

```
% nc why jobID
```

- From the Accelerator GUI, double-click a job and navigate to the **Why** tab.
- From the browser, use the **Jobs in Queue** link from the Workload area of the home page.

The `nc why` command tries to give information about whatever object it is given, whether a job or a file, explaining why the object is in its current state. For example, a job might be waiting for FairShare, or for hardware or software resources. A job could be 'Invalid' because a predecessor dependency has failed, or it has been descheduled after submission, but before it was executed.

The information given as the main reason may not be the only reason a job is waiting. For example, if a job requests both License:foo and Limit:bar, and both are exhausted, it will be hard to tell which is the main wait reason. To save CPU cycles, the NC vovserver stops processing the resources list for additional wait reasons once the first one is encountered.

nc why

Show why a job is in the state it is.

```
nc: Usage Message
NC WHY:
      Show the reason for the current state of the specified job.
```

```
USAGE:
% nc why [OPTIONS] [ID]

OPTIONS:
  -h                -- This help
  -json            -- Format output as JSON (valid for QUEUED or
                   SCHEDULED jobs only).
  -jsondoc         -- Documentation for the output of the -json
                   command-line argument.

EXAMPLES:
% nc why 12345
% nc why -json 12345
% nc why -jsondoc
```

Priority

The scheduling priority affects the order in which the jobs are scheduled. The range is 1 to 15.

Two types of priorities are supported:

- Scheduling priority: Determine the order in which jobs are scheduled. The range is 1(low) to 15(top).
- Execution priority: Influence the execution of the job on the remote machine. The range is 1(low) to 15(top).

There are conditions in which lower priorities supersede higher priorities, such as:

- For the jobs of a given user, higher priority jobs are scheduled before lower priority ones. However, due to the FairShare mechanism, a lower priority job from one user may be dispatched before a higher priority job of another user.
- A low priority job will be dispatched before a high priority job if the resources for the low priority job are available while the resources for the high priority job are not.

The priority of a job may also be used to decide which job can be preempted. Refer to the *Accelerator Administration Guide* for more information about preemption.

Some priority levels have symbolic names, as listed in the following table:

Symbolic Name	Numerical Level
Top	15
High	8
Normal	4
Low	1

In Accelerator, set the priority of a job at submission time with the option -p.

```
% nc run -p high sleep 10
% nc run -p 12 sleep 10
% nc run -p 12.low sleep 10
```

The priority can be set from the GUI using the Retrace Priority Flags dialog from the console. With the command `vsr`, you can use the option `-priority` (which can be abbreviated to `-p`) as shown in the example below:

```
% vsr -p high target      # Use high scheduling priority.
% vsr -p h target        # Abbreviated form.
% vsr -p high.high target # Set both scheduling and execution priority
```

The priorities, in conjunction with the `resources.tcl` file, also affects the amount of parallelism used during retracing:

```
#
# -- This is a fragment of a resources.tcl file.
# -- Typical priority setup.
#
vtk_resourcecemap_set Priority:top      UNLIMITED
vtk_resourcecemap_set Priority:high    50
vtk_resourcecemap_set Priority:normal  10
vtk_resourcecemap_set Priority:low     2

#
# -- This is another example of a resources.tcl file.
# -- Set unlimited parallelism for any level of priority.
# -- However all LOW priority jobs should go to the linux machines.
#
vtk_resourcecemap_set Priority:top      UNLIMITED
vtk_resourcecemap_set Priority:high    UNLIMITED
vtk_resourcecemap_set Priority:normal  UNLIMITED
vtk_resourcecemap_set Priority:low     UNLIMITED  linux
```

This default behavior can be modified with the resource management mechanism. Before a job is dispatched to taskers, its resource list is augmented with one resource representing the priority. The name of the resource is `Priority:xxx`, in which `xxx` represents the selected standard priority level.

The priority-based parallelism can be adjusted by changing the file `resources.tcl`.

The maximum priority that can be assigned by a particular user may be limited by the policy layer. To do so, edit the `policy.tcl` file.

Priorities Relative to Previous Run

When specifying a priority, it is possible to use also the following symbolic values:

Symbolic Name	Meaning
Same	Same priority as before. If not defined, then use low priority.
Incr	Increase previous priority by 1, without exceeding the maximum priority for the user.
Decr	Decrease previous priority by 1, but no less than low priority.


For example:

```
% vsr -p same -set All:drc
% nc rerun -p incr 000123456
```


Execution Priority

The execution priority takes the same values as the scheduling priority. This value is ignored on Windows, while on UNIX it is used in a call to `nice()`. An example of passing this value to `nice()`:

```
niceValue = 8 - executionPriority;
```

 **Note:** For TOP priority (15) `nice(-7)` is called; for LOW priority (1) `nice(7)` is called.

Distributed Parallel Support

Accelerator supports jobs that require multiple CPUs that can be located on different machines.

When a Distributed Parallel (DP) job is submitted to the scheduler, the job is divided into *partial jobs*. Each partial job can require a different set of resources and cumulatively requires all the resources of the Distributed Parallel job. Accelerator schedules each partial job separately. When all partial jobs have been dispatched, one designated partial job executes the actual Distributed Parallel job. Depending on the submission method, there are several ways to take advantage of the computing resources assigned to the other partial jobs.

Submission Methods

There are three common methods to submit Distributed Parallel jobs.

Use -dp to submit a Distributed Parallel job with N partial jobs

This method creates a Distributed Parallel job with two partial jobs. When both partial jobs have been dispatched, the active partial job becomes the master and begins the execution of the command `sleep 10`, while the other partial job waits for the previous partial job to terminate. By default, the active partial job is the first one.

For example:

```
% nc run -dp 2 sleep 10
```

This method is rarely used due to the burden it puts on the tool integrator to find a way to use the other partial jobs.

Use -dp N with vovparallel LSB_HOSTS

When all N partial jobs have been dispatched, the active partial jobs executes `vovparallel`, which sets the environment variable `LSB_HOSTS` before invoking the master command. `LSB_HOSTS` contains the list of all hosts, possibly with repetitions, currently set aside to run all partial jobs of the Distributed Parallel job. The master command is expected to use `rsh` or `ssh` to reach out to those hosts and launch the appropriate software.

In this case, the active component displays the value of the `LSB_HOSTS` environment variable, showing the list of hosts assigned to the DP job group.

For example:

```
% nc run -dp 2 -wl vovparallel lsbhosts printenv LSB_HOSTS
```

Use -dp N together with vovparallel clone

This method clones the specified wrapper script across the selected hosts. The wrapper script then executes the tool-specific commands on the appropriate hosts and will have access to the Distributed Parallel environment variables and job properties.

For example:

```
nc run -dp 2 vovparallel clone mywrapper
```

In this case, the active partial job executes `vovparallel clone . . .` which in turn connects to the other partial jobs to invoke N similar instances of the script `mywrapper` on all partial jobs.

This is the preferred method because it doesn't require `ssh/rsh` functionality to activate the partial jobs, and it allows Accelerator to track memory and CPU utilization for each partial jobs.

At execution time, each `partialTool` job sets some environment variables that help each partial jobs of the Distributed Parallel job understand the role to play. The variables are:

- `VOV_DP_TOPJOBID`, with the `VovId` of the top level Distributed Parallel job
- `VOV_DP_COUNT`, the overall number of partial jobs used for the Distributed Parallel job
- `VOV_DP_RANK`, a number from 1 to `VOV_DP_COUNT` used to identify each partial jobs

Partial Job Rank and Resources

The active partial job, which starts the execution upon completion of dispatching of all partial jobs, is normally component number 1. This can be overridden by using the `dp` active argument.

```
% nc run -dp 8 -dpactive 8 uname -a
```

To define the resource list for each component of the Distributed Parallel job, use the `-dpres` option. This option takes a comma-separated list of simple resources lists. The first element in the list is used for the first partial job, the second element for the second partial job and so on. If there are more partial jobs than elements in the list, the last element in the list is used for all remaining partial jobs.

In the following example, the first partial job is dispatched to a Linux machine, the second partial job to a macosx machine, and all other partial jobs go to UNIX machines. The active partial job is number 2, which is the one running on macosx. The tasker that is running UNIX would becomes the master for the Distributed Parallel job set. This also works with tasker names, or any other tasker resource for that matter (kernel, RAM, etc.).

```
% nc run -dp 10 -dpres "linux,macosx,unix " -dpactive 2 vovparallel clone mywrapper
```

Distributed Parallel Properties

When a Distributed Parallel job is submitted, the `partialTool` job sets the following properties on the top-level job:

DP_ACTIVERANK	The rank of the active partial job, which is the partial job that launches the top-level job.
DP_ATTEMPTS	The number of times the partial jobs have failed to be allocated within the allowed time.
DP_COHORTWAIT	This property is created when <code>-nocohortwait</code> is passed with <code>nc run</code> . This instructs <code>partialTool</code> for each cohort task to finish when its subtask process has

finished rather than wait for the primary job to complete (which is the default behavior). Using `-nocohortwait` sets `DP_COHORTWAIT` to zero (0). Otherwise, the default is 1.

DP_FAILURE	Explanation of failures.
DP_HOSTS	List of hosts for the distributed parallel job . This property is set when the last partial job is launched. The list may contain duplicates.
DP_PORT_X	This is set by partial tool X to the pair host <code>dp_port</code> .
DP_SEMAPHORE	Used by partial jobs to count the rank.
DP_SEMAPHORE2	A second synchronization counter used by partialTool.
DP_SETID	The ID of the set that contains the list of partial jobs.
DP_WAIT	Tells the partial job how long to wait before giving up the slot (default 30 sec).
DP_WAIT_MAX	This is the maximum allowed value for <code>DP_WAIT</code> for a job (default 30 mins).

In addition to these properties on the top-level job, the following property is set on all other jobs:

DP_PART_OF	The top-level job ID.
-------------------	-----------------------

Properties can be accessed via the `vovprop` command or via the Tcl API with `vtk_prop_get`. Once obtained, they can be used in conjunction with `rsh/ssh`, for example, to interact with the chosen hosts.

Using Different Resources and Jobclasses

When specifying DP jobs, you can stack jobclasses so that the primary and component jobs have different resources and job classes.

This is done by setting `VOV_JOB_DESC(dp,resources)` and `VOV_JOB_DESC(dp,jobclasses)` to specify the resources and jobclass labels for the master and subcomponent DP jobs.

For example, two jobclass definitions `mycalibre_a.tcl` and `mycalibre_b.tcl` are defined as follows:

```
:::::::::::::
mycalibre_a.tcl
:::::::::::::
# Copyright (c) 1995-2021, Altair Engineering
# All Rights Reserved.

# $Id: //vov/trunk/src/scripts/jobclass/short.tcl#3 $

set classDescription "My Calibre job resources"

puts "This is mycalibre jobclass"
if { [ info exists VOV_JOB_DESC(dp,resources) ] } {
    set VOV_JOB_DESC(dp,resources) [ string cat $VOV_JOB_DESC(dp,resources) ",RAM/200
CORES/2" ]
}
```


Regardless of the specified wait time, there is a maximum wait time (default 30 mins) that can be specified by manually setting the Distributed `Parallel_WAIT_MAX` property on the job. Use the `P` option with `nc run`, such as

```
-P DP_WAIT_MAX=12.0
```

vovparallel Clone

An example of a script used with `vovparallel clone` is available in `$VOVDIR/training/vnc/simple_dp_script.csh`.

```
#!/bin/csh -f
#
# Example of a script to be used with vovparallel clone.
#
# Example of usage:
# % nc run -dp 4 simple_dp_script.csh
#

set sleepTime = 30
if ( $#argv > 1 ) then
    set sleepTime = $1
endif

# Each application has its own way to determine a rendezvous port.
# In this example, it is a fixed port.
set APPLICATION_PORT = 2345

if ( ! $?VOV_DP_RANK ) then
    echo "ERROR: Variable VOV_DP_RANK not defined."
    echo "      This script needs to be run with vovparallel clone ..."
    exit 0
endif

echo "Hello! I am component $VOV_DP_RANK"

if ( $VOV_DP_RANK == 1 ) then
    echo "This is the master. "
    echo "This should open a socket for communication e.g. $APPLICATION_PORT"
    set DP_HOSTS      = `vovprop GET $VOV_DP_TOPJOBID DP_HOSTS`
    echo $DP_HOSTS
    sleep $sleepTime
else
    echo "This is the tasker"
    set DP_HOSTS      = `vovprop GET $VOV_DP_TOPJOBID DP_HOSTS`
    set masterHost = $DP_HOSTS[1]
    echo "This should communicate with master through ${masterHost}:
$APPLICATION_PORT"
    sleep $sleepTime
endif

exit 0
```

A more advanced script to use with `vovparallel clone` is available in `$VOVDIR/eda/MentorGraphics/vovcalibremt`.

OpenMPI Support

If you have an application that uses OpenMPI, you can submit it as a Distribute Parallel application (options `-dp`, `-dpres`, etc.) and you need to use the wrapper "vovmpirun".

For example, if you want the application to run with N components (possibly on different hosts), you can submit it with:

```
% nc run -dp <N> vovmpirun ./path/to/application
```



Note: This support relies on the fact that OpenMPI uses ssh to start `orted` on the remote hosts. OpenMPI is forced to use `$VOVDIR/hidden_mpi/ssh`.

Multiphase Support

Multiphase support is provided by two additional command arguments to `nc run`:

<code>-multiphase [1 0]</code>	Enables multiphase jobs.
<code>-mpres "resource string"</code>	Sets the resources that will be used for each phase.

The '%' is used as a delimiter for the resources of each phase; for example, `-mpres "linux64 foo%linux64 bar:linux64 baz"`.

In addition, the `autoRescheduleCount` server configuration parameter needs to be set to the max number of job phases or higher. The default is 4, so this applies to jobs with 5 or more phases.

Specifying Resources

By specifying the resources of each phase and designating that certain resources are only allocated to certain taskers, you can run different phases of a job on different taskers. The following options can be set with the `nc run` command:

`mpres RESLIST`

Specification of the resources required by a multiphase job. The RESLIST specifies the resource lists for all phases of the job with % characters delimiting the phases. The sublists of resources for each phase are percent sign delimited.

Example: `-mpres "RAM/200 CORES/2%RAM/20 CORES/3"`

`mpres+ <rsrc>`

Append one resource to the multiphase resource list. This option must follow the `"-mpres"` or `"-mpres<n>"` option, otherwise the resources specified in `"-mpres+"` will be overwritten.

`-mpres1 RESLIST, -mpres2 RESLIST, -mpres<n> RESLIST`

Specify resources for a stage <n> of a multiphase job. The number <n> is in the range from 1 to 9.

Example:

```
-mpres1 "RAM/200"
```

```
-mpres2 "CORES/4 RAM/10"
```

For example: You have two taskers named *tasker1* and *tasker2*. I want to run phase 1 and 3 on *tasker1*, and phase 2 on *tasker2*. Your resources may look like:

```
vtk_resourcecemap_set License:blue UNLIMITED License:blue_tasker1  
vtk_resourcecemap_set License:red UNLIMITED License:red_tasker2  
vtk_resourcecemap_set License:blue_tasker1 1 tasker1  
vtk_resourcecemap_set License:red_tasker2 1 tasker2
```

You could then run a multiphase job as:

```
nc run -multiphase 1 -mpres "linux64 License:blue%linux64 License:red%linux64  
License:blue" -- -e BASE -D /home/jjmcwill/testDir/testMultistage.sh
```

A multiphase job will have two new Job Properties set:

MPRESOURCES	Contains the same resources passed in <code>-mpres</code> , and is used to reset the job resources for each phase.
MPCURRENTPHASE	Contains an integer indicating the current job phase. It starts at one, and has a max value of 9.

The running job script will see an environment variable named `VOV_JOB_PHASE` which is set to the current phase. The script writer will need to use that to decide what work to do for that phase.

- If the script exits with an exit code of 216, Accelerator will increment the job phase, change the job resources, and reschedule the job to run again.
- If the script exits with an exit code of 0, the job is considered "Done", and `MPCURRENTPHASE` is reset to 1.

Failed Jobs

If a job fails during a phase with a code other than 0 or 216, it is considered `FAILED` and `MPCURRENTPHASE` will not increment. If the job is invalidated and re-run (such as, `nc rerun -f JOBID`), the job will re-run starting at `MPCURRENTPHASE` and further phases will run if the job exits with code 216, as described above.

Logging

After the first phase is run, subsequent phases of the job will have the command rewritten so that the wrappers are passed `-a -A`, telling the wrappers to append to the job log. This is so that all phases of the job get their stdout and stderr logged to the same file. If this was not done, each phase of the job would overwrite the log, and you would only see the output from the last phase that was run.

If Accelerator does not detect one of the standard vov wrappers at the beginning of the command line, it will assume the command is not using a wrapper. In this case, it will look for the standard `> ;` redirect symbol in the command and replace it with `>> ;`.

REST Support

In the payload for submitting a job via REST, two new fields are allowed: `multiphase` and `mpres`. Setting `multiphase = True` enables multiphase job support. Setting the `mpres` field behaves the same as described for the command line argument

described above. Re-running a multiphase job that has failed via the REST re-run API will behave similarly to rerunning a failed multiphase job from the command line as described above.

Job Submission Arguments

Job submission can be affected by the value of the optional variables `NC_RUN_ARGS` and `NC_RUN_ARGS_AFTER`, which specify a list of arguments that are pre-pended and appended to the argument list passed to the submission command.

For example, if the variables are defined as follows:

```
% setenv NC_RUN_ARGS "-D"  
% setenv NC_RUN_ARGS_AFTER "-jobproj myproj123"
```

Then the submission

```
% nc run -p high sleep 10
```

Becomes effectively

```
% nc run -D -p high -jobproj myproj123 sleep 10
```


Modify Running Jobs

Interactive Jobs

Interactive jobs require attention as they run, whereas batch jobs are run unattended.

Interactive jobs are only supported on UNIX in Accelerator. There are three types of interactive jobs, which are described in the table below.

Interactive Job	Example	Option	Example
Job that requires a X display.	<code>xterm -e vi abc</code>	-Ix	<code>% nc run -Ix xterm -e vi abc</code>
Job that requires a TTY with remote control of signal dispatching by means of Ctrl-C and Ctrl-Z.	<code>bash</code>	-Ir	<code>% nc run -Ir bash</code>
Job that requires a TTY (stdin, stdout) but keep local control of signal dispatching by means of Ctrl-C and Ctrl-Z. These are jobs where you want redirect stdin to a file.	n/a	-Il	<code>% date nc run -Il tr '[a-z]' '[A-Z]'</code>

You can limit the number of interactive jobs that can run concurrently, both at the global and user level. This is accomplished by creating a limit resource and setting it as the interactive job limit in the `vnrcrun.config.tcl` file. For example:

```
set VOV_JOB_DESC(interactive,limit) Limit:interactive
```

Or, for a per-user limit:

```
set VOV_JOB_DESC(interactive,limit) Limit:interactive_@USER@
```

The resource must exist prior to adding these lines to the file.

Use the -splitsderr Option

Use the `-splitsderr` option to write the `stderr` output of the interactive job to `stderr`. The default is to write the job's `stderr` output to `stdout`. Note that using this option will probably result in garbled terminal output due to intermingling of `stdout` and `stderr` outputs.

When to Use -Il and -Ir

If you use the option `-Ir`, then handling Ctrl-C and Ctrl-Z are done **remotely** on the remote host where the job is running. Use `-Ir` to interact with the job.

If you use the option `-Il`, then handling `Ctrl-C` and `Ctrl-Z` are done **locally** with the submission shell. Use `-Il` to redirect the stdout of the job to a file or a pipe.

Interactive Job Logs

Logging is supported for interactive job. For example, the following command will produce a transcript in `mylog.txt`:

```
% nc run -I -l mylog.txt - /bin/bash
```

Options to Use with Interactive Job Logging	Conditions
<code>-I</code> option	Only use for jobs that require attention as they run. <code>-I</code> should not be used in scripted jobs where the intent is to capture output. In general, each user should never have more than one or two interactive jobs running concurrently.
<code>-wl</code> option	Intended for when the job is launched at a terminal and real time logging is required. <code>-wl</code> is preferred over <code>-I</code> when only output tailing is required. A pseudo terminal server is not employed (keyboard input is not processed) and the job is more robust over network and window glitches.
<code>-w</code> option	Preferred method for blocking a job and capturing its output in a script. The job can be issued with a <code>-w</code> option. This blocks until the job completes. The log can then be accessed via <code>nc info -l</code> .



Note: Neither `-wl` or `-I` should be used in scripts where there is no controlling terminal providing supervision.

For further information, please submit a support request at [Altair Community](#).

Jobs That Require a DISPLAY: Option `-Ix`

To run a graphical tool interactively, use the option `-Ix` with `nc run`. This option adds the component `+D(DISPLAY=$DISPLAY)` to the job environment.

- To use this option, the `DISPLAY` environment variable must be set for the display to refer to the host that you want to view.
- If `DISPLAY` does not contain a hostname component, such as `"unix:0.0"` or `":0.0"`, then `nc run` command substitutes the hostname of the submission host. You must set a `nc run` value containing a hostname component to display the windows on a host other than the submission host.

For most graphical tools, all interactions occur through the windows and no terminal is needed. Batch jobs, and those started with only the `-Ix` option, do not have a pty allocated.

There are tools, such as Cadence NanoRoute and some simulators, which expect to have the regular streams connected to a pty, and will not operate properly (that is, just exit) unless there is a pty. For such jobs, use the `-Ir` or `-Il` option to ensure a pty is allocated.

For tools such as simulators that interpret the INT (interrupt) signal, typically ^C, to stop the simulation and return to interactive control, you may need to start in an xterm (x terminal) to gain full functionality. In this case, how to submit the job is similar to the following example:

```
% nc run -Ix xterm -e vsim -do ...
```

Modify Scheduled Jobs

The `nc modify` command allows modifying fields in the scheduled jobs.

nc modify

```
vnc: Usage Message

NC MODIFY:
  Modify scheduled jobs

USAGE:
  % nc modify [OPTIONS] [jobId] ...           (operate on job with that id)
  % nc modify [OPTIONS] [!] ...             (operate on most recent job)
  % nc modify [OPTIONS] [-set setName] ...   (operate on all jobs in setName)
  % nc modify [OPTIONS] [-selrule rule] ...  (operate on all jobs selected by rule)

OPTIONS:
  -h                               -- This help.
  -v                               -- Increase verbosity.
  -showfields                       -- Show fields that can be modified.
  -<FIELDNAME> <NEWVALUE>         -- Set the specified field to the
                                   specified value.
  -changegrab <RESMAP> [-]<N>      -- Change the quantity of a resourcemap
                                   grabbed for a running job. May not
                                   combine with other options.

EXAMPLES:
  % nc modify -jobclass short 0012345
  % nc modify -res License:xxx 0012345
  % nc modify -jobname superman 0012345
  % nc modify -res License:xxx -set MySet
  % nc modify -group /time/users !
  % nc modify -jpp smallest -numa pack 0012345
  % nc modify -changegrab Limit:foo -1 -selrule 'user=mary AND resources~foo'
```

wx modify

```
wx: Usage Message

WX MODIFY:
  Modify scheduled jobs
```

```
USAGE:
    % wx modify [OPTIONS] [jobId] ...
                                     (operate on job with that id)
    % wx modify [OPTIONS] [!] ...
                                     (operate on most recent job)
    % wx modify [OPTIONS] [-set setName] ...
                                     (operate on all jobs in setName)
    % wx modify [OPTIONS] [-selrule rule] ...
                                     (operate on all jobs selected by rule)

OPTIONS:
    -h                                -- This help.
    -v                                -- Increase verbosity.
    -showfields                       -- Show fields that can be modified.
    -<FIELDNAME> <NEWVALUE>         -- Set the specified field to the
                                     specified value.
    -changegrab <RESMAP> [-]<N>      -- Change the quantity of a resourcemap
                                     grabbed for a running job. May not
                                     combine with other options.

EXAMPLES:
    % wx modify -jobclass short 0012345
    % wx modify -res License:xxx 0012345
    % wx modify -jobname superman 0012345
    % wx modify -res License:xxx -set MySet
    % wx modify -group /time/users !
    % wx modify -jpp smallest -numa pack 0012345
    % wx modify -changegrab Limit:foo -1 -selrule 'user=mary AND resources~foo'
```

To see a list of the fields that can be modified, use the `-showfields` option as shown below:

```
autoflow
autoforget
autokill
cmd
deadline
dir
env
fstokens
group
jobclass
jobname
jobproj
jpp
legalexit
nojournal
numa
preemptable
priority
res
res,aux
scheddate
submithost
systemjob
tool
xdur
xpriority
```

Restrictions and Consequences

The following fields can be changed any time, including when the job is running.

autokill	Duration job runs before being killed automatically
fstokens	FairShare tokens; when changed, moves the job to the proper FairShare bucket
xpriority	Execution priority
jobname	Can only be changed by job's owner
jpp	Job placement policy
legalexit	If the job is not running, the exit status is checked again with the new legal values; if it does not match, the job is invalidated
numa	Job placement non-uniform memory access policy; Linux only
preemptable	Flag indicating the job can be preempted
priority	Schedule priority
systemjob	Flag that indicates it is a system job; not normally modified this way
xdur	Expected duration

The following fields cannot be changed while the job is running. They can be changed by the job's owner or an administrator.

autoflow	Flag that indicates a job should be skipped; if changed, job is moved to proper bucket
nojournal	Flag that turns off journal entries for the job
res	Resources; if changed for a scheduled job, the job is re-queued
scheddate	Date/time job was scheduled

The following fields cannot be changed while the job is running. They can only be changed by the job's owner (not an administrator).

autoforget	Flag that indicates the job will be forgotten after completion
cmd	Command line of job; job is invalidated if changed
deadline	The desired date/time the job should be completed
dir	File system path where job runs; job is invalidated if changed
env	Named environment of job; job is invalidated if changed
group	FairShare group; if changed for a scheduled job, the job is re-queued

jobclass	Resource group; if changed for a scheduled job, the job is re-queued
jobproj	Project associated with job; if changed for a scheduled job, the job is re-queued
res,aux	Aux resources; if changed for a scheduled job, the job is re-queued
submithost	Host from which job is submitted
tool	Name of tool associated with command

Modifying job fields will be restricted if a VovUserGroup named System:jobmodify exists. If this VovUserGroup exists, only users who are a member of the group will be able to modify job fields. Users not in the VovUserGroup will receive an authorization error.

FairShare Groups

In Accelerator, FairShare groups are managed by either the information in the `vnc.swd` directory that contains the `policy.tcl` file, or the `vovfsgroup` utility. Every user has a default FairShare group, which is set in the `policy.tcl` file. Use `nc run` with the option `-g` to select a different group. An error occurs if the user specifies a non-existent group or a group to which the user does not belong.

For example:

```
% nc cmd vovshow -users
% nc run -g /time/regression sleep 10
% nc run -g xxxx sleep 10
nc: USER ERROR: No such group defined: 'xxxx'
```

FairShare Subgroups


Subgroups can be specified by using the `-sg` option. Subgroups can be created at submit time as opposed to groups, which must be defined ahead of time. Subgroups allow a user to allocate shares of computing resources to subsets of their own workload.

For example:

```
% nc run -sg subgroup sleep 10 (/time/users.john:subgroup)
% nc run -g /time/regression -sg subgroup sleep 10 (/time/regression.john:subgroup)
```

Job Placement Policies

Accelerator supports multiple *job placement policies*: methods to choose on which tasker to run a job.

 **Note:** These policies are advisory only. Some job scheduling scenarios will be handled by the scheduler with overrides that ignore the user-specified job placement policy.

Job Placement Policy	Description
fastest	This is the default job placement policy: among all the taskers that can execute a job, choose the one with highest power. The assumption is that a tasker with higher power can complete the job faster.
slowest	Among all the taskers that can execute a job, choose the tasker with the least power. This policy may be useful to run regression jobs on older, less powerful hardware.
first	As soon as the scheduler finds a tasker to execute the job, it uses that tasker without checking all other taskers. This policy is useful for lowering the scheduling effort.
smallest	Among all the taskers that can execute a job, choose the tasker with the least amount of unused slots and unused RAM ((MB of unused RAM) + 16000*(Number of unused cores)). This policy tends to pack the jobs on machines that are already busy, thus keeping idle machines available if a large job is submitted.
smallram	Among all the taskers that can execute a job, choose the tasker with the least amount of total RAM. This policy is useful to pack the jobs on the smaller machines first, which keeps large machines available if a large job is submitted.
largest	Among all the taskers that can execute a job, choose the tasker with the most amount of unused slots and unused RAM ((MB of unused RAM) + 16000*(Number of unused cores)). This policy tends to spread the jobs on idle machines. In most cases, this policy may not be the most effective.

Accelerator also supports three CPU-affinity policies for machines that have a [NUMA architecture](#). These policies apply to jobs after they are placed on a specific tasker. This is for Linux only.

CPU-Affinity Policy	Description
pack	NUMA control: assign the job to a NUMA node with the least number of available resources that will fit the job. If none of the NUMA nodes have sufficient job slots and RAM, the job will be allowed to run on as many NUMA nodes as needed to satisfy its resource requirement.
spread	NUMA control: assign the job to a NUMA node that has the largest number of available resources.
none	NUMA control: allow Linux to place jobs. The Linux CPUs Allowed affinity list will be all the CPUs on the system (default).

Choose the Job Placement Policy

At submission time, the option `-jpp` can be used to specify, at most, one of the job placement policies and one CPU-Affinity policy. The list is comma-separated. If multiple conflicting policies are specified, the last policy on the list will be used.

To place jobs on the same machines, use `first` or `smallest`.

```
% nc run -jpp slowest ./my_not_so_important_job
% nc run -jpp slowest,spread ./my_not_so_important_job
% nc run -jpp smallest,pack ./my_job
% nc run -jpp smallram ./my_job
```

In a job class, the value of `VOV_JOB_DESC` (`jpp`) can be set:

```
# Fragment of a jobclass definition:
set VOV_JOB_DESC(jpp) "smallest,pack"
```

Clean Up Log Files

All log files are normally stored under the subdirectory `./vnc_logs`. To remove all obsolete log files in the current working directory, use the `nc clean` command.

nc clean

This command cleans up obsolete log files and environment files that have been generated by jobs submitted to the scheduler.

```
nc: Usage Message

NC CLEAN:
  This command cleans up obsolete log files and environment files
  that have been generated by jobs submitted to the scheduler.
  By default the command cleans the current working directory
  (i.e. removes logs and environment files of the jobs executed in the
  current working directory).

  If a list of directories is provided, the command will clean up
  the files in those directories instead.

USAGE:
  % nc clean [OPTIONS] [LIST_OF_DIRS]

OPTIONS:
  -deep N      -- Clean the jobs from all directories in which the user
                has executed jobs in the past N days. The directories
                are found from the journals.
  -dir <dir>   -- Specify additional paths to check.
  -h           -- Help usage message.
  -nozap       -- Do not 'zap' isolated nodes. Allows the cleaning of the
                current directory to proceed faster.
  -P PERIOD    -- Install a periodic job to run the cleaning automatically.
  -R           -- Clean the directories recursively.
  -v           -- Increase verbosity.
  -zap         -- Do 'zap' of isolated nodes (see man vsz for more info).

EXAMPLES:
  % nc clean -h
  % nc clean
  % nc clean -dir /tools/logs/VNC_LOGS -dir /scratch/logs/VNC
  % nc clean . /tools/logs/VNC_LOGS /scratch/logs/VNC
  % nc clean -zap
  % nc clean -deep 10
  % nc clean -deep 3 -P 3d
```

Comments

Use the option `-R` (recursive) to also clean up the subdirectories.

```
% nc clean -R
```

From within scripts, it is recommended to use the option `-nozap`, which tells `nc clean` to skip the calling of the zapping utility `vsz`, which can be expensive in terms of time and load on the server.

```
% nc clean -nozap
```

If you do not remember the directories where you have run jobs, you can use the deep cleaning option `-deep` that automatically looks in the journals to find out all the directories in which jobs have been run. This option accepts an integer parameter that specifies the number of days to go back in the journals. The following example will go back 10 days:

```
% nc clean -deep 10
```

To have Accelerator automatically run `nc clean` every day, schedule a periodic job. For example, the following command schedules a cleanup once a day in the current directory:

```
% nc clean -P 1d
```

Debug Jobs without Running Accelerator

On occasion, jobs that run successfully outside of Accelerator fail when run through Accelerator. When this occurs, mostly likely the setups are not the same: the environment, inputs or other parameters may be different, a misconfiguration or there is a problem with NFS.

To resolve such issues, using the command `nc debug` can show you the steps that Accelerator takes to run the job.

When some jobs are not behaving as expected, use the command `nc debug jobId` to get the steps that Accelerator uses to run the job.

nc debug

```
vnc: Usage Message
```

```
NC DEBUG:
```

```
If a job appears to behave differently when executed by NC  
than when it runs without using NC, you can use this command  
to debug the problem.
```

```
The command gives you the step-by-step description of what  
NC does to run the job, so that you can do  
the same thing without going through NC.
```

```
For example, if you find a job runs fine without NC, but fails  
in NC, it might simply be that the environment is not set correctly.  
By following the steps provided by this command, you will be able  
to determine what is wrong.
```

```
USAGE:
```

```
% nc debug <jobId>
```

```
OPTIONS:
```

```
-h                - Show this message
```

Debug Jobs Example

Following the steps in the example below, modified or as is, you can check if you are running the same job in the same setup as it would be in Accelerator.

By eliminating vovserver and vovtasker from the picture, it very often becomes obvious or easy to figure out what the problem is. Sometimes it is a missing environment variable. Sometimes it is an NFS problem, etc. In the unlikely event that run the same job successfully following these steps, there might be something missing or wrong in how Accelerator runs the job, or something is misconfigured.

Example:

```
% nc debug 01597942  
  
# This job was run on host bear. To run the same
```

```
# job without going through Accelerator, please follow these steps:

# 1. Logon to the machine (if necessary)
rsh bear -l john      ; # or ssh bear

# 2. Change to the directory
cd /home/john

# 3. Switch to the environment
ves SYNOPSIS

# 4. Run the job without wrappers or redirection
./myscript input1 input2
```

Job Runtime - Monitor and Profile

When a job is running through a vovtasker, the tasker automatically monitors RAM and CPU utilization of the job, including all of its children.

Job statistics are sampled about once a minute. This data sampling rate does not capture jobs that complete in less time than the sampling period.

The MAXRAM is expressed in Megabytes (MB), where 1MB = 1<<20 bits (left-shift decimal "1" 20 times is the binary equivalent of 1 million). The CPU time is stored in ms (milliseconds), but is expressed in s (seconds).

CPU Progress and Run Status Indicators

Accelerator monitors CPU and RAM utilization for all the running jobs. The CPU utilization information is available in four fields:

CPUTIME	The total accumulated CPU time in milliseconds.
CPUPROGRESS	Percentage of CPU accumulated in the unit time. For example, if in 60 seconds a job uses 60 seconds of CPU time, then the CPUPROGRESS is going to be 100. This field can be 0 (zero) for jobs that are stuck: holding onto the CPU resource but not running, which makes the CPU unavailable for other jobs. This field can also be greater than 100 for multi-threaded jobs.
LASTCPUPROGRESS	A timestamp indicating the last time CPU usage has increased. This is used to identify stuck jobs.
RUNSTATUS	A descriptive text field that shows how well the job is doing. Some typical values are Good, Paging, NoCpu. The complete list of values is shown below.

Table 1: Values of the RUNSTATUS Field

n/a	Insufficient information to determine CPU progress. Typical for jobs that have just started.
Good	The progress is greater than 70%
Medium	Progress is between 10% and 70%
Poor	Less than 10% CPU utilization, but no swapping of pages.
Paging	The progress is less than 10% and the job is swapping at a rate greater than 1000 pages per second.
NoCpu	The job is not accumulating any CPU time.
Susp	The job is suspended.

Job Profiling

When job profiling is activated, Accelerator tracks and plots performance statistics over the time the job is running.

The profiling plots show, in order, the following performance data over time:

1. RAM usage
2. VM size
3. CPU utilization
4. Cumulative Read I/O
5. Cumulative Write I/O
6. License checkouts (one plot per license)

The output of job profiling is a set of plots as shown below:

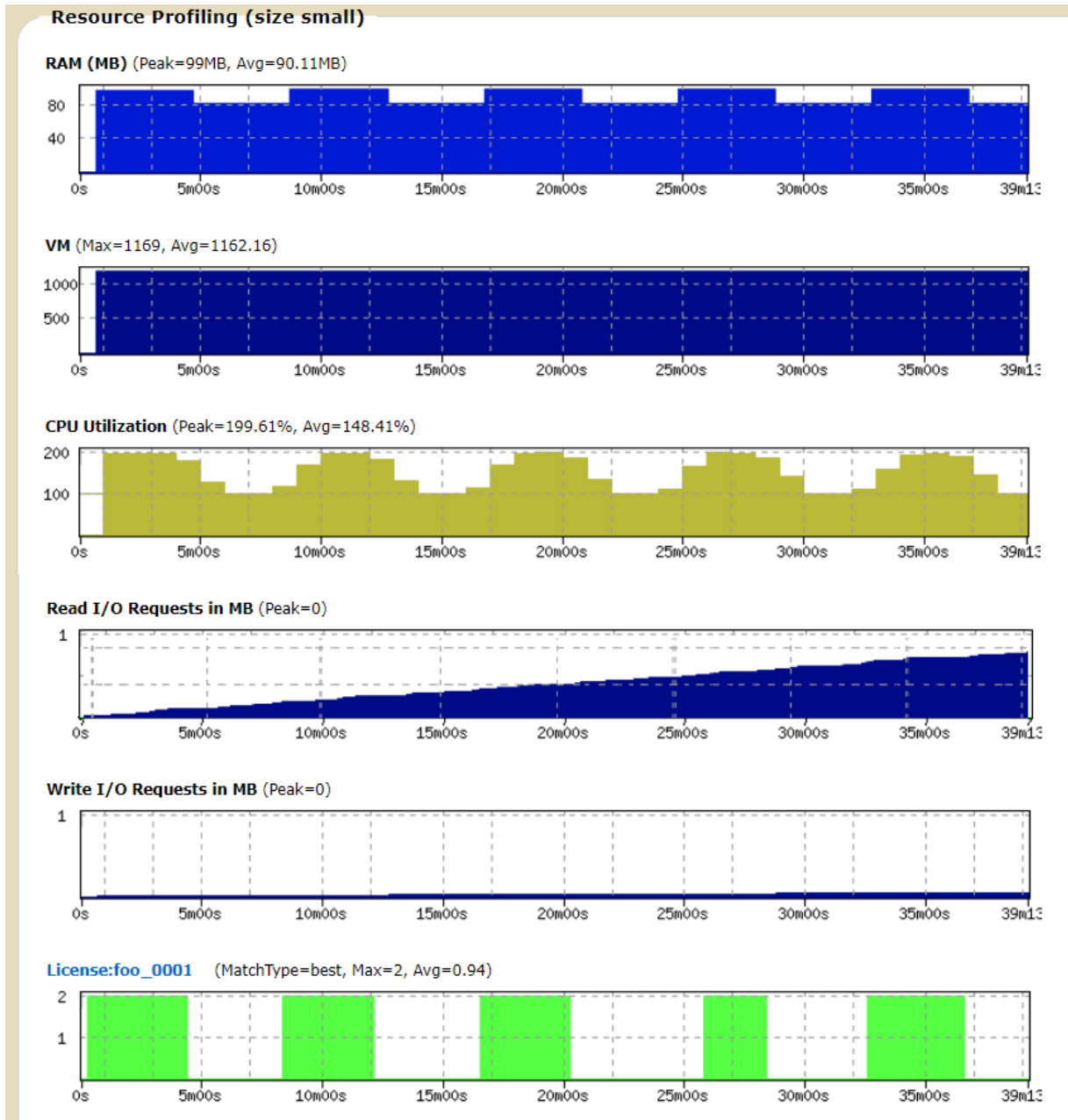


Figure 2:

To activate profiling on a single job, use the option `-profile` of `nc run` as shown below:

```
% nc run -profile myJob
```

To view a profile, use the browser interface and visit the specific page for the job.

To activate job profiling for a jobclass, set the following:

```
# In a job class definition  
set VOV_JOB_DESC(profile) 1
```


To activate job profiling for all jobs, use the file `$VOVDIR/local/vncrun.config.tcl` and add a line like this:

```
# In the file $VOVDIR/local/vncrun.config.tcl
...
set VOV_JOB_DESC(profile) 1
...
```

Job I/O Profiling

The I/O job profiling feature enables Accelerator to track and plot performance statistics over the time the job is running. A summary of this information is displayed after the job completes by use of the `nc info -ioprofile JOBID` command.

This feature can only be activated with a Mistral license, which must be installed at `$VOVDIR/local/mistral.dat`. Check with your system administrator for accessing and installing the license. You can get more information about the Mistral license at [Altair License Management](#).

The summary information shows I/O performance metrics broken down by filesystem. The statistics are aggregated from all processes in a job. The statistics reported include:

- **Data Xferred** – The total number of data transferred in the specified I/O direction.
- **Throughput** – A data rate calculated as a ratio of total data transferred divided by job run time.
- **Effective BW** - A data rate calculated as a ratio of total data transferred divided by total latency.
- **I/O Ops** – The number of I/O operations issued by the job for the specified I/O direction.
- **Total Latency** – The elapsed time for all I/O system calls, summed.
- **Latency/Op** - A ratio of total latency and number of I/O operations for the specified I/O direction

```
I/O Profiling Results - Read
Filesystem      Data Xfered      Throughput      Effective BW      I/O Ops      Total
Latency  Latency/Op
/dev           1048576000B      349525333B/s    11351MB/s         1000
 92372us       92us/op
/users/kfeind  0B               0B/s            N/A                0              0us
              N/A

I/O Profiling Results - Write
Filesystem      Data Xfered      Throughput      Effective BW      I/O Ops      Total
Latency  Latency/Op
/dev           0B               0B/s            N/A                0              0us
              N/A
/users/kfeind  1048576000B      349525333B/s    500MB/s           1000
2094136us     2094us/op

Filesystems Accessed
Filesystem      Type      Source
/dev            devtmpfs devtmpfs
/users/kfeind   nfs       sdc-storenad01:/export/user_home/kfeind
```


To activate profiling on a single job, use the option `-profile` of `nc run` as shown below:

```
% nc run -iopprofile ./myjob
```

To view the summary profiling statistics, invoke the following CLI command after the job has completed.

```
% nc info -iopprofile JOBID
```

To view a graphical display of I/O time series statistics while the job is running, invoke the `nc gui` client as follows.

 **Note:** This is a preview feature.


```
% nc gui -iopprofile JOBID
```

Monitor the Workload

List Jobs

To show the status of jobs recently submitted, use `nc list`.

The default is to show up to 20 jobs submitted by the user running `nc list`. With some options, described below, you may also view jobs belonging to other users.

 **Note:** This command can impose a significant load on the system. Please review the help info below for suggestions how to obtain job info efficiently.

The Accelerator Administrator may have configured methods to mitigate this load, including caching.

`nc list`

List jobs currently in the system.

```
nc: Usage Message

NC LIST:
    List jobs currently in the system.
    The list is ordered by increasing job id,
    normally the same as the submission order.

    The behavior can be controlled by $VOVDIR/local/vnclist.config.tcl
    and by the variables
    NC_LIST_FORMAT
    NC_LIST_CACHE_DIR
    NC_LIST_CACHE_TIMEOUT

NOTE ON CACHES and MORE EFFICIENT METHODS:
    This command may use local, client-side caches.
    Caches are activated by setting NCLIST(cache,enable) to 1
    in the file $VOVDIR/local/vnclist.config.tcl

    These caches can significantly reduce
    the load on the scheduler in the case of repeated calls.
    The default timeout for these caches is 30s.

    There are better ways to get information about jobs, especially in
    scripts. Please consider the following efficient methods:
    % nc getfield $JOBID ...
      -- To get specific info about a job
    % nc cmd vovset count SETNAME ...
      -- To count jobs in sets by status
    % nc wait ...
      -- To block waiting for jobs to complete

USAGE:
```

```
% nc list [OPTIONS]
```

OPTIONS:

```
-O <format>          -- Specify a different Output format.
                       Refer to the manual for a description
                       of formats. For experts.
                       This can also be specified via the NC_LIST_FORMAT
                       environment variable.

-O+ <format>         -- Add one or more fields to the default output
                       format.

-H                   -- When used with -O, show header line.

-l                   -- Long format: show group, user, host,
                       and full command.

-L                   -- Very long format: show start and end dates,
                       duration.

-c                   -- Count jobs: this option affects the output
                       format.
                       It adds a column with the the position of each
                       job in the list of jobs to be shown.

-S <rule>
-select <rule>
-selrule <rule>     -- 3 ways to specify the selection rule. For
                       experts. The clause 'isjob' is added to the
                       selection rule.

-S+ <rule>          -- Alternate (OR) selrule; jobs that match any
                       selrule will be shown. The clause 'isjob' is
                       added to the selection rule.

-subjobs            -- Show also sub-jobs too
                       (e.g. jobresumer, partialtool)

-systemjobs        -- Show also system-jobs

-alljobs           -- Show all types of jobs,
                       including sub-jobs and system-jobs.

-a                 -- Show jobs for all users.

-r                 -- Show only running jobs.

-f                 -- Show only failed jobs.

-s                 -- Show only suspended jobs.

-q                 -- Show only queued jobs.

-u <user>          -- Show jobs belonging to given user. Ignored if
                       used with -selrule or -set option.

-first <index>     -- Show jobs starting at index. By default,
                       first index is 1.

-last <index>      -- Show jobs ending at index. By default, last
                       index is -1, which is the last job.

<num>              -- Show first <num> jobs if <num> is positive.
                       Show last <num> jobs if <num> is negative number

-J <jobName>       -- Show only jobs with given job name.
                       WARNING: this option can place a high load on
                       the Accelerator server in large workload
                       environments due to the need to compare strings
                       for each job, it is recommended to avoid calling
                       this unless truly required.

-set <setName>     -- Show only jobs in the given set.
                       This option can be repeated to show the content
                       of multiple sets. If a job belongs to multiple
                       sets, it may be reported more than once.

-dir <directory>  -- Show only jobs in the given directory.

-cache             -- Use result cache. It is recommended to utilize
                       the result cache to reduce the load on the
                       Accelerator server
                       in cases where list queries are scripted.
                       By default, the cache expires after 30s.

-v                 -- Increase verbosity.
```

```
-h                -- Print brief usage.

EXAMPLES:
% nc list
% nc list -c
% nc list -a -l
% nc list -O "@USER@ @GROUP@ @DURATION@"
                    -selrule "duration>60"
% nc list -H -O "@USER@ @GROUP@ @DURATION@"
                    -selrule "statusnc==Failed"
% nc list -dir .
% nc list -a -r -s
                    (Show all running and suspended jobs)
% nc list -selrule "duration>600 statusnc==Running"
% nc list -first 10 -last 20
% nc list 5
% nc list -10
```

Summary of All Jobs

The command `nc summary` is used to show a short summary of jobs in the system.

```
nc: Usage Message

NC SUMMARY:
  Get a summary report for all of my jobs.

USAGE:
  % nc summary [options]

OPTIONS:
-a, -all                -- Print report for all users.
-all_users            -- Same as -a
-all_sets             -- Show all sets.
-b                   -- Show buckets.
-h                   -- Help usage message.
-P                   -- Print report for all projects.
-p PROJECT           -- Print report for given project (repeatable).
-set SETNAME        -- Show report for just that set.
-u USER            -- Print report for given user (repeatable).
-w                   -- Show detailed info about wait reasons.
```

Following is an example of the output of `nc summary`:

```
% nc summary -all
Altair Accelerator Summary For All Users
TOTAL JOBS      2101      Duration: 6h33m
Done            2005
Queued          95
Running         1

JOBS  GROUP  USER  TOOL  WAITING FOR...
50  groupA  john  vtclsh  ' License:fintronic#1'
45  groupB  mary  vtclsh  ' License:fintronic#1'
```

To view the summary of jobs for a specific user, use option `-u name` with the command `nc summary`:

```
% nc summary -u john
Altair Accelerator Summary For User john
TOTAL JOBS      1101      Duration: 2h30m
Done            2005
Queued          50
Running         0

JOBS  GROUP  USER  TOOL      WAITING FOR...
50   groupA  john   vtclsh    ' License:fintronic#1'
```

Notification of Job Status

The Accelerator `vovnotifyd` notification daemon accesses the server's event stream and then sends a notification for jobs that request it.

To enable this notification, the `MAILTO` property must be set: use the option `-m` or `-M` option with the `nc run` command. An example is shown below:

```
% nc run -m sleep 10
% nc run -M ":ERROR" simulate chip.spi
```

The format of the property of `MAILTO` can be configured as follows:

```
recipientList
recipientList : verbList
recipientList : ALL
: verbList
```

`recipientList` is the list of the e-mail recipients. `verbList` is the list of verbs for which notifications must be sent. The supported verbs are listed below.

```
DESCCHEDULE - Job has been dequeued.
DISPATCH   - Job has left the queue and has been routed to an execution host.
ERROR       - Job has exited with a failure.
FORGET      - Job has been forgotten.
RESUME      - Job has been resumed.
STOP        - Job has exited successfully.
SUSPEND     - Job has been suspended.
```

If the `recipientList` is empty, a notification is sent to the owner of the job. If the `verbList` is empty, then a notification is sent only when the job terminates.

For example:

```
john : ERROR - Send mail to the user 'john' if the job terminates in error.
: STOP ERROR - Send mail to the job owner when the job terminates.
john mary: ALL - Send mail to the users 'john' and 'mary' for anything that happens
to the job.
```

Change the MAILTO Property After Job Submission

To change the MAILTO property, use the `vovprop` utility. The following are examples of getting, setting, and deleting the property:

```
% nc cmd vovprop get 000012345 MAILTO
% nc cmd vovprop set -text 000012345 MAILTO "mary : STOP ERROR"
% nc cmd vovprop delete 000012345 MAILTO
```

Invoke the GUI

Job execution can be monitored with `nc gui`.

This command opens a monitoring tool; no interactive capabilities (such as configuration or running jobs) are provided. Interactive capabilities are available with `nc cmd vovconsole`.

nc gui

Show a grid view of the jobs in a specified set.

```
nc: Usage Message

NC GUI:
  Show a grid view of the jobs in a specified set.
USAGE:
  % nc gui [OPTIONS] &
OPTIONS:
  With no options, the GUI shows all jobs of the current
  user.

  -all
  -a                -- Show all jobs.
  -u <user>        -- Show jobs for specified user.
  -s <SETNAME>
  -set <SETNAME>
  -setname <SETNAME> -- Show specified set.
  -timeout <TIMESPEC> -- Stop async update after this time (default 2h).
  -submit
  -limitGui <N>    -- Override the limit of 3 max GUI per user.

  -batch <file>    -- Execute specified file after the GUI is ready

  -metrics
  -metricsConfig <file> -- Use specified metrics configuration file.
  -taskers
  -fontsize <size>   -- Specify the normal font size. Default is 10.
                    Legal range is 3 to 36.

  -title <title>   -- Choose title of X11 window.
  -ioprofile <jobId> -- Show job I/O profiling timeseries statistics
                    plots. The job must have been submitted with
                    the -ioprofile option. (preview feature)
```

EXAMPLES:

```

% nc gui & -- Show all my jobs
% nc gui -all & -- Show all jobs.
% nc gui -set SomeSetName -- Show specified set.

% nc gui -submit -- Job submission dialog.
% nc gui -limitGui 5 -- Allow you to run up to 5 "nc gui" (default 3)

% nc gui -metrics & -- Show the scheduler metrics.
    
```

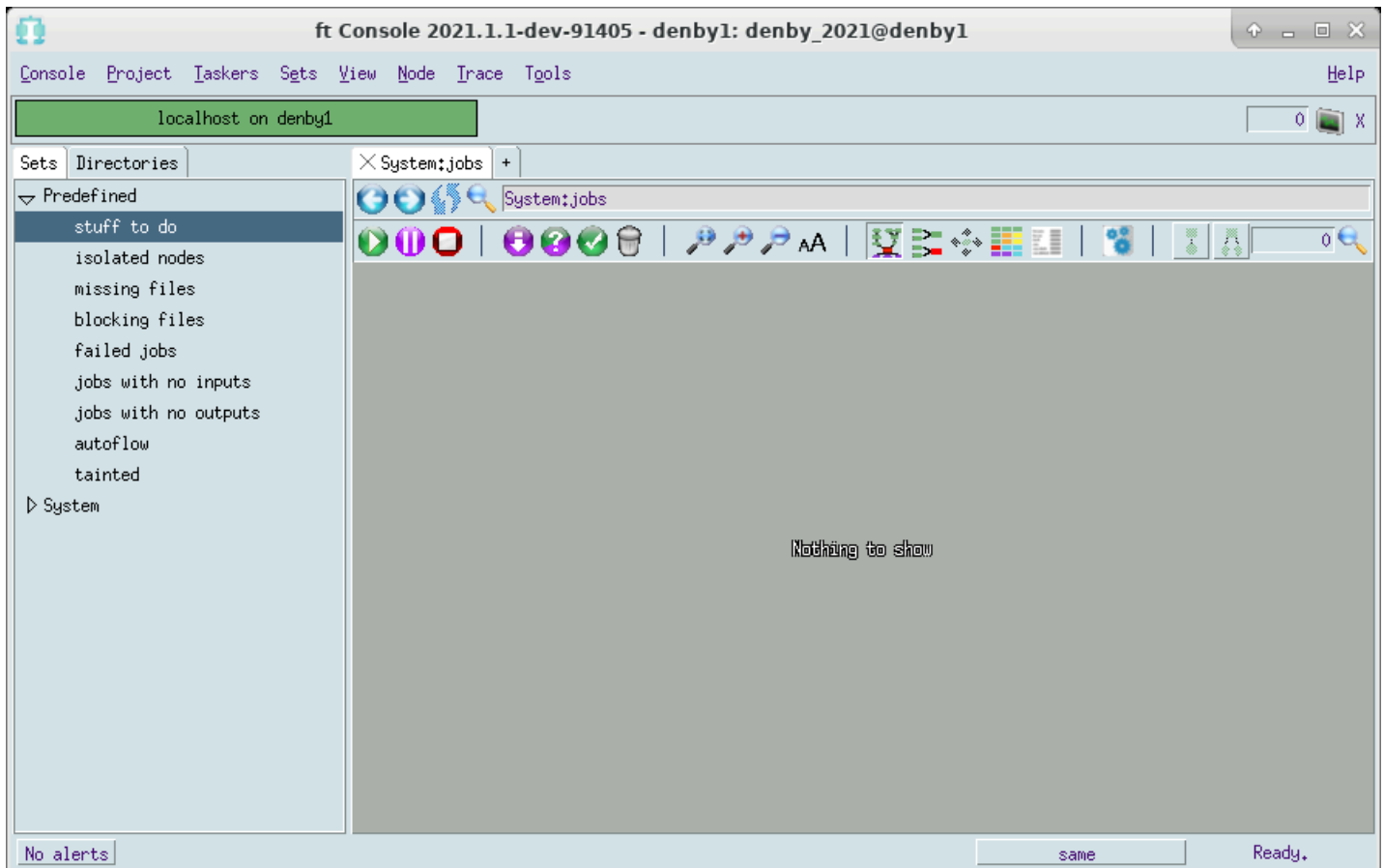






















Figure 3: GUI that opens after entering nc cmd vorconsole &

Icons

All icons provide descriptions that you can find by hovering over the icon.

	<p>Run/Rerun</p>	<p>Update the current object, whether a set or a node. All jobs necessary to bring the object up to date are scheduled for execution. The jobs that can run now start running.</p>
---	------------------	--

	Stop/Dequeue	Un-schedule a job that is not yet running or abort a retrace request. Running jobs are unharmed and keep running. Graceful Stop. Cyan/asked-to-run jobs will turn back purple/invalid.
	Stop/Dequeue	Stop a running job or a retrace request. This does the same thing as the dequeue above, but in addition kills the selected running jobs. It's a forced stop. Running jobs that have been stopped will turn red.
	Navigate Backward	Display the previous set that you were browsing.
	Navigate Forward	Display the next set that you were browsing, when you already went backward.
	Refresh	Recompute the current set based on the selection rule.
	Find	Finds files or jobs in the trace. There are two identical Find icons. The left one close to the name of the set being displayed triggers changing the setname box into a search box for the current graph. The other on the extreme right brings up a new dialog that lets you search for files or jobs.
	Invalidate	Invalidates the selected nodes.
	Try Validate	Tries to validate the selected jobs and its downcone.
	Force Validate	Equivalent of make -t.
	Forget	Removes the job/file from the graph of dependencies. This does NOT remove the file from disk. It just removes it from the dependencies allowing blocked jobs to start. Usually rerunning <code>ovbuild</code> is enough to bring the dependency back.
	Fit	Reduces/expands the graph so that it fits in the window.
	Zoom In	Expands the graph to better see some nodes.
	Zoom Out	Reduces the graph to see more nodes.
	Select Font Size	Reduce or enlarge the size of the characters in the nodes.

	Vertical View	Display the current set using the "graph" widget.
	Horizontal View	Display the current set using the "horizontal" widget.
	Grid View	Display the current set using the "grid" widget.
	Stats View	Display the current set using the "stats" widget.
	Graph Settings	Set the various preferences for Weight Driven Placement. This takes you to the Graph tab of the Preferences dialog.

Show the Hosts/Taskers

The command `nc hosts` shows the list of the hardware resources currently connected to an instance of Accelerator. These hardware resources are called "taskers" in Accelerator.

```
nc: Usage Message

NC HOSTS
  Show taskers that are currently in the cluster along with tasker metadata.

  The default output includes:
  NAME LOAD STATUS RUN/SUSP SLOTS HEARTBEAT RESERVATIONS MESSAGE

  Each tasker takes on the name of its host by default.

  The "RUN/SUSP" column shows running jobs and suspended jobs,
  respectively.

  The "SLOTS" column shows total job slots.

  The heartbeat is the age of the most recent heartbeat received by the
  vovserver for that specific tasker.

  The reservations column shows shorthand representations for who or what
  the tasker is reserved and the time remaining for the reservation. The
  shorthand format is TYPE:NAME, where TYPE is one of:
  G (group), I (ID), B (bucket), C (jobclass), P (project), or U (user).

USAGE:
  % nc hosts [OPTIONS]

OPTIONS:
  -a                -- Show all known hosts (used with -m).
  -ALL              -- Show resources for each tasker.
  -c                -- Show consumable resources (e.g. RAM and CPUs).
  -f                -- Show list of tasker fields.
  -h                -- Help usage message.
```

```
-hw <HW>          -- Show only taskers that match HW constraints.
-INFO             -- Same as -O ...fields about host, arch, model, ...
-LOAD            -- Same as -O ...selection of fields about load...
-m              -- Show machine parameters (RAM, CPUfreq, ...)
-O <fmt>         -- Specify output format. The format string can contain
                  elements like @FIELDNAME@ or @FIELDNAME:WIDTH@ where
                  a negative width means left-align and a positive width
                  means right-align.

-r              -- Show status and resources for each tasker.
-rl             -- Show resources (legacy: pre-2013.03 format).
-RAM            -- Same as -O "@NAME@ RAM/@RAM@ RAMFREE#@RAMFREE@
                  RAMTOTAL#@RAMTOTAL@"

-rule <SELRULE> -- Show only taskers that match the given selection rule.
                  Use "vovselect fieldname from taskers" for the complete
                  list of fields that can be used in the rule.
                  Example rules:
                      "status==READY"
                      "status!=OVRLD slots>8"
                  Can accept multiple constraints.
-SLOTS          -- Same as -O "@NAME@ SLOTS/@SLOTS@ SLOTSTOTAL#@SLOTSTOTAL@
                  CORES/@CORES@ CORESTOTAL#@CORESTOTAL@"
-slowdown       -- Used only for testing.
```

EXAMPLES:

```
% nc hosts
% nc hosts -m
% nc hosts -a -m
% nc hosts -hw 'RAMTOTAL>18000'
% nc hosts -f
% nc hosts -O "RAMFREE#@RAMFREE@ SWAP/@SWAP@ M=@MODEL@"
% nc hosts -O "@I:4@ @NAME:-14@ @STATUS:-8@ @HOST@ "
% nc hosts -RAM
% nc hosts -hw 'RAMTOTAL>18000' -RAM
% nc hosts -ALL | grep -A8 ^lnx001
% nc hosts -rule "cores>4 ramtotal<20000" -O "@name@ @corestotal@"
```

An example is shown below:

```
% nc hosts
# TASKER          LOAD STATUS   JOBS  MESSAGE
1 alpaca          0.01 ready    0/1   Workstation idle
2 bison           0.07 ready    0/1   Workstation idle
3 blue-srv        0.15 ready    0/1
4 cayman          0.00 susp     0/1   Off hour tasker (will start at 19:00)
5 cheetah         0.00 ready    0/1   Workstation idle
6 comet-srv       0.23 ready    0/1
7 everett         0.00 ready    0/2
8 jupiter-srv    0.07 ready    0/2
9 mars-srv        0.06 ready    0/2
10 moon-srv       0.09 ready    0/1

% nc hosts -r
...
% nc hosts -m
...
```

Use vovselect for Querying

The `nc hosts` command can be used for querying, but it can sometimes take several minutes to return results, which causes some nodes to show up as "N/A". `nc hosts` will query the server and return significant amounts of data, but the server loading will directly affect the response time of the command.

In order to avoid such delay, you can use `vovselect` to run the query, as it prefilters the output server-side before returning it to the client.

Use the table below to understand the mapping of fields between the `nc hosts` and `vovselect` commands.

nc hosts	vovselect from TASKERS	vovselect from HOSTS
ARCH	ARCH	ARCH
CAPABILITIES	CAPABILITIES	NA
CAPACITY	CAPACITY	CPUS
CLASSRESOURCES	CLASSRESOURCES	NA
CLOCK	CLOCK	CPUCLOCK
COEFF	COEFF	NA
CONSUMABLES	CONSUMABLES	NA
CORES	CORES_AVAIL	NA
CORES_AVAIL	CORES_AVAIL	NA
CORESTOTAL	CORESTOTAL	CPUS
CORESUSED	CORESUSED	NA
CPUS	CPUS	CPUS
CURLOAD	CURLOAD	NA
DOEXEC	DOEXEC	NA
DONETINFO	DONETINFO	NA
DOPROCINFO	DOPROCINFO	NA
DORTTRACING	DORTTRACING	NA
EFFLOAD	NA	NA
EXTRAS	EXTRAS	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
FULLINFO	FULLINFO	NA
GROUP	GROUP	NA
HB	NA	NA
HBPP	NA	NA
HEARTBEAT	HEARTBEAT	NA
HOST	HOST	NAME
ID	ID	NA
IDINT	IDINT	NA
LASTJOBID	NA	NA
LASTUPDATE	LASTUPDATE	NA
LIFETIMEJOBS	LIFETIMEJOBS	NA
LOAD1	NA	NA
LOAD15	NA	NA
LOAD5	NA	NA
LOADEFF	NA	NA
MACHINE	MACHINE	MACHINE
MANUALPOWER	NA	NA
MAXLOAD	MAXLOAD	NA
MESSAGE	MESSAGE	NA
MESSAGESYS	MESSAGESYS	NA
MESSAGEUSER	MESSAGEUSER	NA
MODEL	MODEL	NA
NAME	NAME	NAME
NUMJOBS	NA	NA
OSCLASS	OSCLASS	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
PERCENT	PERCENT	NA
PERSISTENT	PERSISTENT	NA
PID	PID	NA
POWER	POWER	NA
RAM	RAM	NA
RAMFREE	RAMFREE	NA
RAMTOTAL	RAMTOTAL	RAMTOTAL
RAWPOWER	NA	NA
RELEASE	RELEASE	NA
RESERVEDBY	RESERVEDBY	NA
RESERVEEND	RESERVEEND	NA
RESERVEFORBUCKETID	RESERVEFORBUCKETID	NA
RESERVEFORID	RESERVEFORID	NA
RESERVEGROUP	RESERVEGROUP	NA
RESERVEJOBCLASS	RESERVEJOBCLASS	NA
RESERVEJOBPROJ	RESERVEJOBPROJ	NA
RESERVEOSGROUP	RESERVEOSGROUP	NA
RESERVESTART	RESERVESTART	NA
RESERVEUSER	RESERVEUSER	NA
RESOURCECMD	RESOURCECMD	NA
RESOURCES	NA	NA
RESOURCESEXTRA	NA	NA
RESOURCESPEC	RESOURCESPEC	NA
RUNNINGJOBS	RUNNINGJOBS	NA
SLOTS	NA	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
SLOTSTOTAL	SLOTSTOTAL	NA
STATSREJECTCORES	STATSREJECTCORES	NA
STATSREJECTOTHER	STATSREJECTOTHER	NA
STATSREJECTRAM	STATSREJECTRAM	NA
STATSREJECTRESERVED	STATSREJECTRESERVED	NA
STATSREJECTSLOTS	STATSREJECTSLOTS	NA
STATSVISITS	NA	NA
STATUS	NA	NA
SWAP	SWAP	NA
SWAPFREE	SWAPFREE	NA
SWAPTOTAL	SWAPTOTAL	NA
TASKERGROUP	TASKER	NA
TASKERNAME	TASKERNAME	NAME
TASKERSLOTSSUSPENDABLE	TASKERSLOTSSUSPENDABLE	NA
TASKERSLOTSSUSPENDED	TASKERSLOTSSUSPENDED	NA
TASKERSLOTSUSED	TASKERSLOTSUSED	NA
TASKERTYPE	TASKERTYPE	NA
TIMELEFT	TIMELEFT	NA
TMP	TMP	NA
TYPE	TYPE	NA
UPTIME	NA	NA
UPTIMEPP	UPTIMEPP	NA
USER	USER	NA
VERSION	VERSION	NA
VOVVERSION	VOVVERSION	NA

Monitor Jobs, Taskers and Resources

The activity of Accelerator can be monitored with a dialog.

The dialog is invoked with:

```
% nc monitor
```

The following is a list of the tabs available in the dialog:

TaskersGroups	The activity of tasker groups.
Taskers	The activity of taskers.
Taskers HW	The hardware offered by taskers.
Taskers Resources	The resources offered by taskers.
Who	Who is running jobs.
Running Jobs	The progress of running jobs.
Running Commands	The details of running commands.
Running Details	The details of running jobs.
Resources	The usage and availability of resources.
Queued Jobs	The jobs in the job queue.
Queue Buckets	The jobs in the job queue organized by groups of similar jobs (called 'buckets').
FairShare	The FairShare statistics.

TaskerGroups	Taskers	Tasker HW	Tasker Resources	Who	Running Jobs	Running Commands	Running Details	Resources	Queued Jobs	Queue Buckets	FairShare
Type:Name	Total	InUse	Rsrvd	ooQ	Available	%Util.	MapsTo				
1 License:AL002_HMPoll	unlim	0	0	0	unlim						
2 Limit:user0265_justo	1	0	0	0	1	99.12%					
3 License:AL002_HFSoli	100000	0	0	0	100000	0.00%					
4 License:AL003_HMPSD_	unlim	0	0	0	unlim						
5 Limit:user0418_justo	1	0	0	0	1	99.79%					
6 Limit:user0354_justo	1	0	0	0	1	97.64%					
7 Limit:user0100_justo	1	0	0	0	1	98.94%					
8 License:AL003_HMPSD_	unlim	0	0	0	unlim						
9 License:SolverNode	100000	0	0	0	100000	0.00%					
10 Limit:user1312_justo	1	0	0	0	1	98.16%					
11 License:AL003_HMPano	unlim	0	0	0	unlim						
12 License:AL003_SIMLAB	100000	0	0	0	100000	0.00%					
13 License:AL001_HMActi	unlim	0	0	0	unlim						

Figure 4:

Statistical Information about Resources and Jobs

Reports provide statistical information about all the jobs that are run during a specified time and the usage of the resources that run jobs. The resources include CPU time, licenses, memory and more.

Statistics

Statistics can be used to determine if additional resources are required, if resources could be better utilized by rescheduling jobs, and if resources are excessive.

Job reports can be viewed on a browser interface.

`host:port/cgi/jobstats.cgi` generates a report on all the jobs executed in a given time interval. This report includes the average and maximum duration of the jobs, and the average and maximum waiting time.

	Time		Jobs		Duration		WaitTime		MaxRAM (MB)		CPU Time	
	TZ	Start	Tot.	Fail	Avg	Max	Avg	Max	Avg	Max	Avg	Max
1	PST	00:00:00	555		36m36s	1d00h	1d03h	3d10h	5	14	2s	1m50s
2	PST	01:00:00	631		42m36s	2d07h	1d00h	3d11h	5	14	2s	5m24s
3	PST	02:00:00	1,477		20m18s	2d13h	7h19m	3d11h	6	14	1s	4m37s
4	PST	03:00:00	558		51m47s	1d10h	23h17m	3d13h	6	14	3s	2m37s
5	PST	04:00:00	599		40m32s	3d10h	23h30m	3d14h	5	14	2s	7m42s
6	PST	05:00:00	366		38m46s	1d12h	22h03m	3d14h	6	14	2s	2m38s
Σ			4,186		avg-38m26s	max-3d10h	avg-21h14m	max-3d14h		max-14		max-7m42s


Showing 6 out of 6 rows | Limit rows to display: 

Figure 5:

Resource Statistics

The Resource Statistics page generates a report of all resources used in a given time interval.

The utilization of the various resources and their criticality are shown below.

Resource Statistics For 5h43m													
Fri Feb 21 00:00:00 PST 2020 - Fri Feb 21 05:42:59 PST 2020													
#	Resource	Total		Used			Queued			Unmet Demand			
		Avg	Max	Avg	Max	%	Avg	Max	TotTime	Avg	Max	%	TotTime
1	License:Xcelium_Single_Core	1.00	1	0.92	1	92.36	20135.47	20259	13y50d	18592.02	20259	92.33	12y48d
2	Group:ies_paleturquoise_aqua_random	∞	∞	1.16	3	0.00	10609.40	14136	6y336d	0.00	0	0.00	0s
3	Group:ies_seagreen_palevioletred_dfq_regression	∞	∞	1.09	5	0.00	0.13	4	45m15s	0.00	0	0.00	0s
4	Group:ies_paleturquoise_peachpuff_random	∞	∞	4.53	9	0.00	962.66	1062	229d06h	0.00	0	0.00	0s
5	Group:ies_paleturquoise_zbf_black_random	∞	∞	2.50	3	0.00	2124.60	2277	1y141d	0.00	0	0.00	0s
6	Group:ies_powderblue_zbf_random	∞	∞	21.64	24	0.00	583.79	599	139d01h	0.00	0	0.00	0s
7	Group:ies_sienna_dvz_random	∞	∞	0.40	8	0.00	0.75	9	4h15m	0.00	0	0.00	0s
8	Group:ies_powderblue_regression	∞	∞	21.14	24	0.00	309.06	344	73d14h	0.00	0	0.00	0s
9	Group:ies_powderblue_sip_regression	∞	∞	0.00	1	0.00	0.06	1	20m54s	0.00	0	0.00	0s
10	Group:ies_paleturquoise_xsr_random	∞	∞	2.27	5	0.00	2200.78	2247	1y159d	0.00	0	0.00	0s
11	Group:small_paleturquoise	∞	∞	17.82	22	0.00	0.03	2	8m35s	0.00	0	0.00	0s
12	Group:ies_sienna_alm_regression	∞	∞	0.01	3	0.00	0.23	3	1h19m	0.00	0	0.00	0s
13	Group:ies_paleturquoise_aqua_cvm_random	∞	∞	0.98	4	0.00	3471.18	3675	2y96d	0.00	0	0.00	0s
14	Group:ies_seagreen_yellowgreen_salmon_random	∞	∞	0.40	31	0.00	8.28	132	1d23h	0.00	0	0.00	0s
15	Limit:sanity_cmodel_segdv	17777.00	17777	2.00	2	0.01	0.00	0	0s	0.00	0	0.00	0s
16	Group:ies_paleturquoise_xau_regression	∞	∞	5.00	6	0.00	100.69	130	23d23h	0.00	0	0.00	0s
17	Group:ies_paleturquoise_comflowerblue_random	∞	∞	4.99	8	0.00	1572.62	1620	1y09d	0.00	0	0.00	0s
18	Limit:regression_notool_segdv	17777.00	17777	7.32	11	0.04	600.91	754	143d03h	0.00	0	0.00	0s
19	Group:ies_seagreen_dodgerblue_peachpuff_random	∞	∞	0.09	10	0.00	1.45	31	8h15m	0.00	0	0.00	0s
20	Group:ies_seagreen_tad_forestgreen_random	∞	∞	0.24	1	0.00	168.41	207	40d02h	0.00	0	0.00	0s

Figure 6:

To view the plot of the utilization of a specific resource, click the **graph** button that is next to the resource name in the Resource column of the Resource Statistics page.

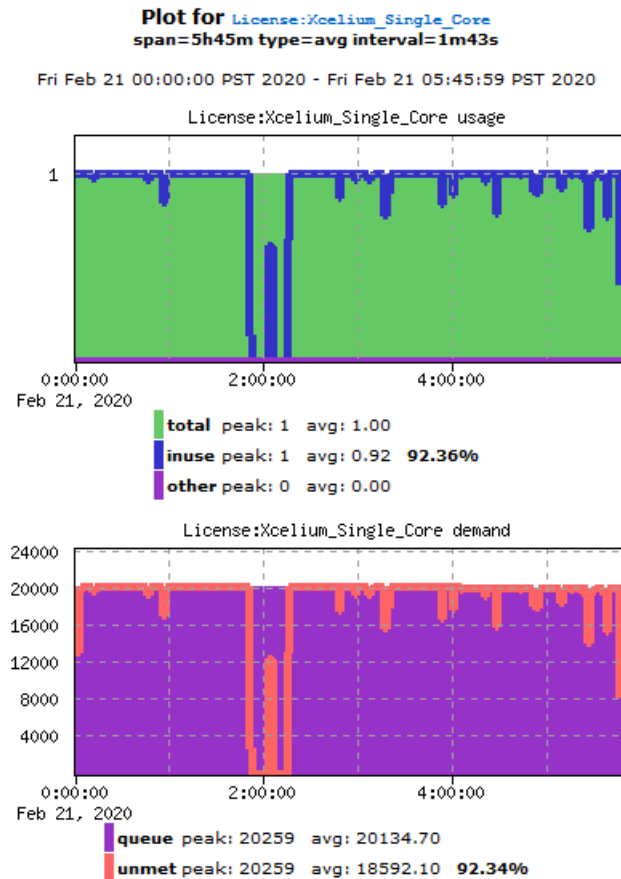


Figure 7:

To view the detailed report of the utilization of a specific resource, click the desired resource name in the Resource column of the Resource Statistics page.

Resource Map: Group:ies_seagreen_palevioletred_dfq_regression

Owned by (server) (rank=0)
Preemption Method: NONE

Total	unlimited
Available	unlimited
In use	1
Resource Reservations	none
Used by others	0
Utilization	170.01%
Technical Details: ilr=0 r=0 flags=	

[Get Report For All Resources](#)

[Get Graph For This Resource](#)

[Find Jobs Using This Resource](#)

New Resource Map Reservation

For User:

For FairShare Group:

For OS Group:

For JobClass:

For JobProject:

For Job ID:

Start Now

Select Duration

ID	262905496
Maps to	
Feature	
Expiration date	never
Left to expire	never expire
Last event	Sat Jan 11 00:20:48 2020
Last use date	Fri Feb 21 02:17:20 2020

Note: This is the most recent matching of jobs to license handles, which is delayed by as much as the cycle of LicenseMonitor compared to the information above.

Resource User			VOV Info				License Mgr. Info						
#	User	Host	Matching Status	Project	Job Id	Job Status	Age	Ask	Got	License Server	Handle	Age	Match Type
1	taylor	cen69b1	n/a(1)		517371329	RETRACING	2d22h	1					

1 matched resources total. Up to 100 can be shown.

Figure 8:

Resource Plots

Altair Accelerator tools save information about the utilization of each resource map, which can be displayed in graphical form. Two plots are generated for each resource: one plot shows the utilization of the resource; one plot shows the demand for the resource.

The plot can be generated over a period of 1, 2, 4, or 24 hours. Options are available to zoom in or out of the view, and to pan the view. You can focus on the details of the selected range of time as shown below.

To view a plot, on the browser go to the Resource Statistics page, and then press the **graph** button that is next to the desired resource name in the Resource column. The Resource Statistics page is shown in [Statistical Information about Resources and Jobs](#).

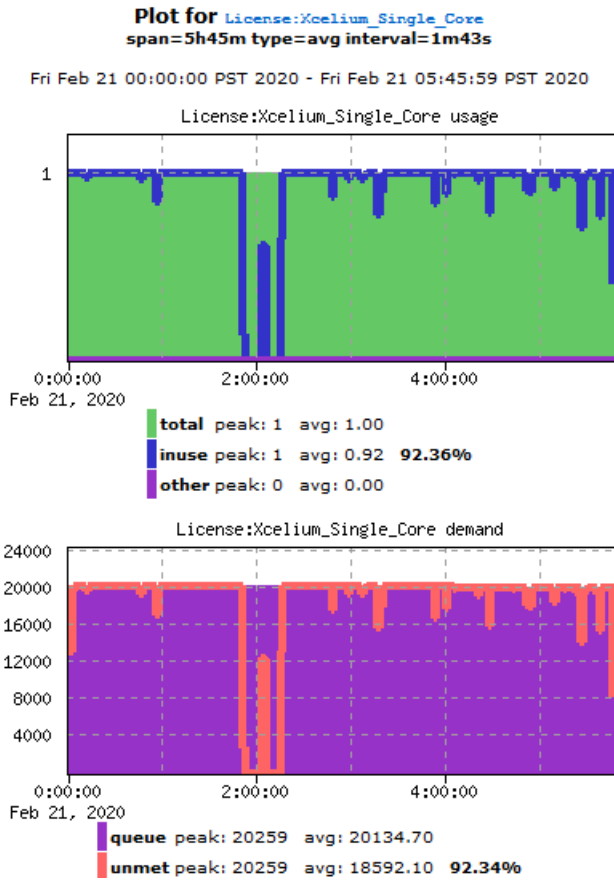


Figure 9:

Resource Utilization

The first set of plots shows the availability (green background) and utilization (blue line) of a resource in an interval of today > (24 hours). The height of the blue line indicates the usage of the resources: the higher the value of the plot line, the more efficient is the usage of the resources.

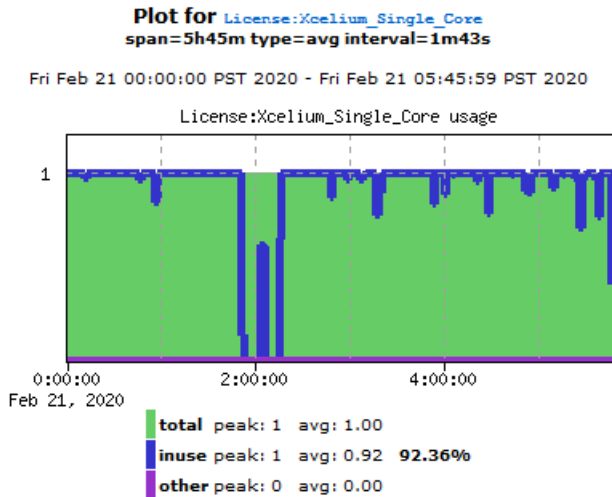


Figure 10:

Queued Jobs and Unmet Demand

The second set of plots show the demand for the given resource. In this example, it starts with no jobs in the queue. After a few minutes jobs are started. More jobs are added as previously submitted jobs are executed. about 350 jobs is added. More jobs from the are added as submitted jobs are executed. (In this example, short jobs are submitted.)

Accelerator can relate the jobs in the queue with the utilization of a resource. When a resource is exhausted and more jobs in the queue ask for that same resource, that indicates *unmet demand*. Unmet demand is represented by the red line in the plot.

When the red line tracks the dark blue line, it indicates the resource is critical: this condition limits the ability of FlowTracer to process the queued jobs faster. When the red line is significantly below the dark blue line, it indicates that the dispatching of the queued jobs is limited. The limitation could be caused by the insufficiency of another resource such as CPUs.

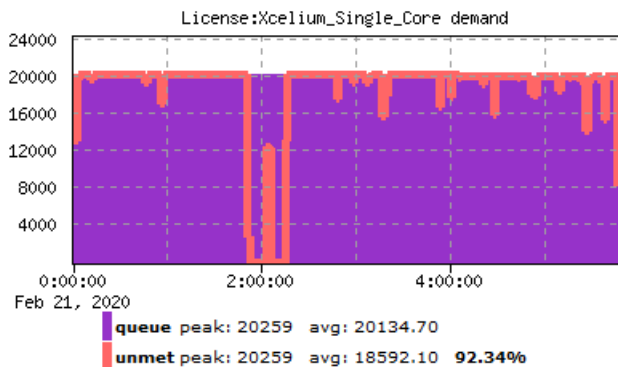



Figure 11:

Use the Plots to Plan for Future Software/Hardware Purchases

If the plotted resource represents an expensive software license, it is important to look carefully at the *unmet demand* curve.

If the unmet demand is low, presumably there is an excess capacity for this license and the performance of the queueing system would be the same even if this resource was reduced. On the other hand, if the unmet demand is high, purchasing more licenses could be a benefit.

 **Note:** When resource utilization is low, it may indicate there is a bottleneck in the queue and that unmet demand is actually high. For example, there may be a rush-hour of jobs that exceeds the availability of licenses, which then limits the usage of other resources. These situations typically appear as spikes in the graph. The queuing system manages the spikes of license demands. However, for planning purposes, it is important to analyze these spikes and determine if there is a significant effect on project timelines as well as resource utilization.

Job Resource Plots

Information about the project jobs can be viewed through the browser on the Job Plots page.

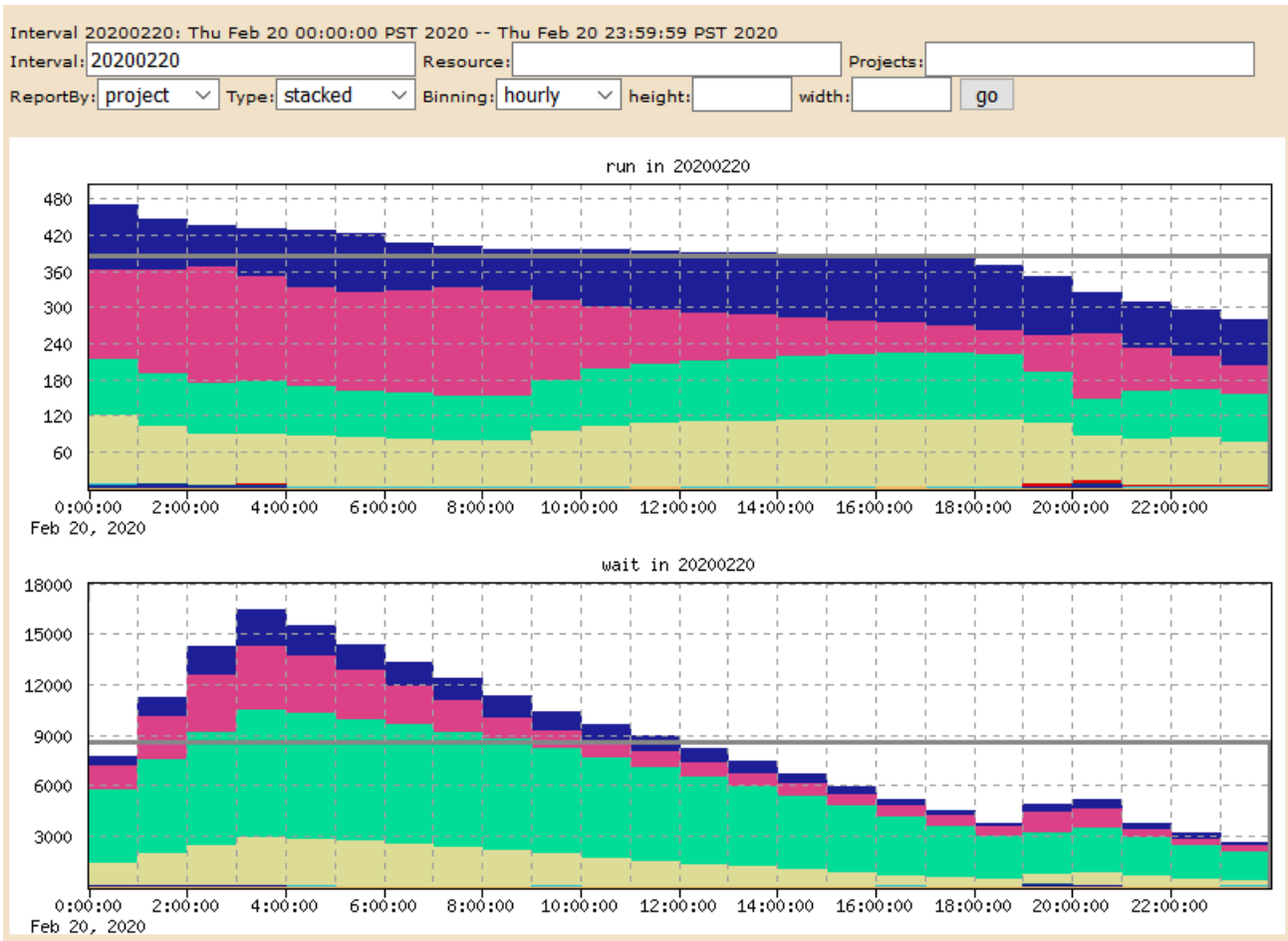
There are two plots: one plot displays the number of jobs per project that are running; the other plot displays the number of jobs per project that are waiting (in queue). Each project is defined by color, which is listed in the table at the bottom of the page. Other details, including the number of jobs per project that are running and waiting, and specific details about the Running and Waiting information are included in the table.

By default, all jobs projects are stacked: displayed together in a plot, with hourly intervals (binning). The options of information to display:

- Report by: User, host, jobclass or project. The default setting is project.
- Type: Stacked or separated. The default setting is separated.
- Binning: None, minutely, hourly, daily, weekly, monthly. The default setting is hourly.

To configure the view, select the desired item per drop menu and then press the **go** button.

An example of the projects stacked is shown below:



			Running Info					Waiting Info			
	project	Jobs	Peak	Avg	TotalTime	AvgRun	Peak	Avg	TotalTime	AvgWait	
1		3	1	0.00	5m13s	1m44s	0	0.00	0s	0s	
2	eris	232	42	0.61	14h38m	3m47s	93	2.81	2d19h	17m24s	
3	herse	716	31	0.53	12h50m	1m04s	140	5.95	5d22h	11m58s	
4	io	15	1	0.01	15m34s	1m02s	1	0.02	24m49s	1m39s	
5	kafka	1,440	1	0.05	1h17m	3s	1	0.00	10s	0s	
6	kore	13	9	1.10	1d02h	2h02m	2	0.07	1h35m	7m19s	
7	leda	10	1	0.01	12m53s	1m17s	1	0.04	52m28s	5m14s	
8	metis	140	16	0.10	2h24m	1m01s	27	0.13	3h00m	1m17s	
9	moon	3,582	129	94.41	94d09h	37m57s	2,972	1444.17	3y349d	9h40m	
10	neptune	8,985	113	89.11	89d02h	14m16s	7,714	4830.69	13y85d	12h54m	
∑		29,126	479	386.07	1y21d	19m05s	17,174	8617.44	23y222d	7h06m	

Showing 10 out of 14 rows | Limit rows to display: | Show all rows | Filter ignore case

Figure 12:

An example of the jobs separated is shown below:

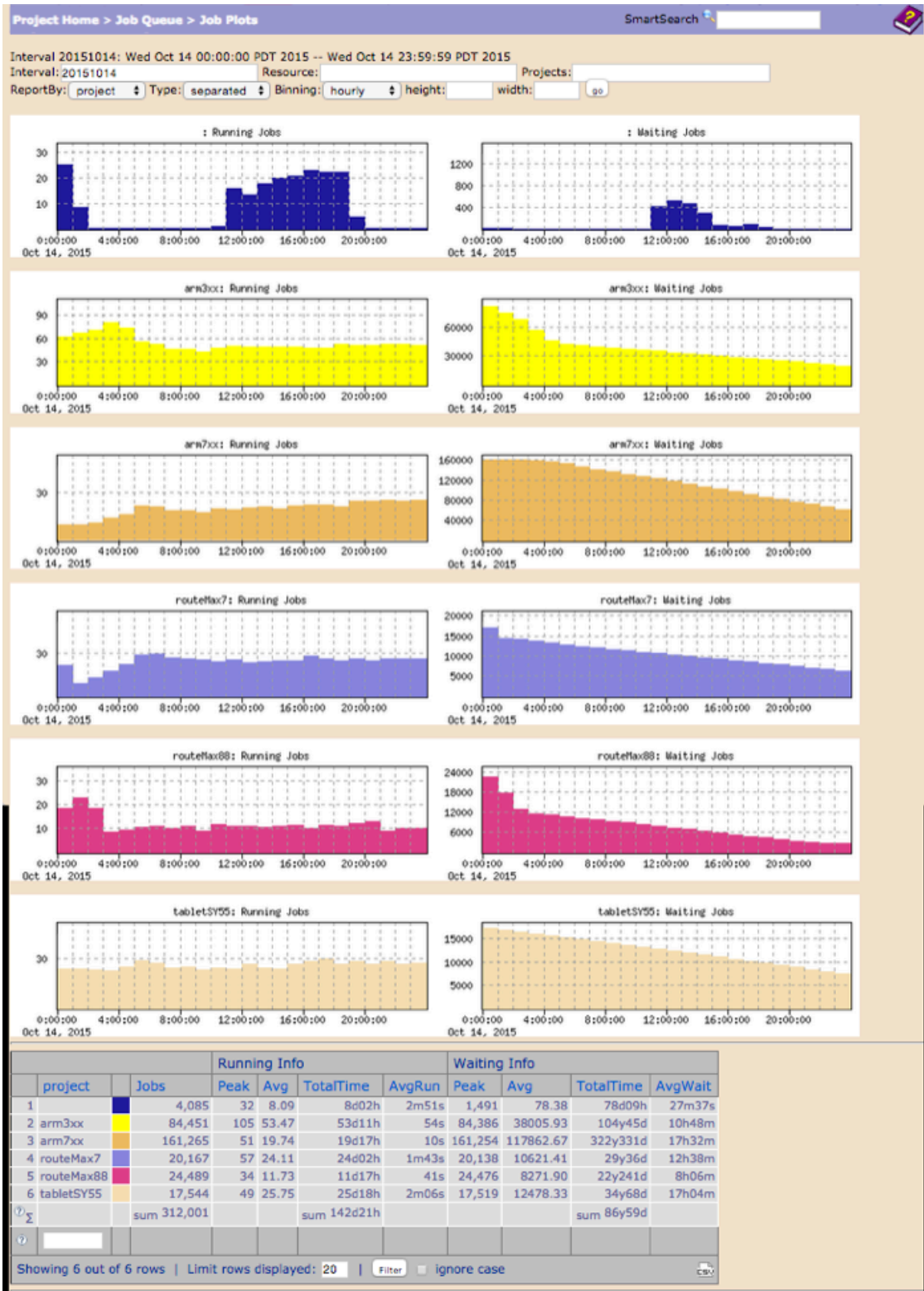



Figure 13:

Environment Control

Setting the environment is critical for correct job execution. Accelerator provides two methods to control the execution environment.


1. Use a **snapshot of the environment** used at submission time.

This method is the simplest and is automatically selected if the environment variable `VOV_ENV` is not defined. The disadvantage of this method is that the snapshot may not be portable across platforms.

 **Note:** This method is not available for Windows.

2. Use a **named environment**, which allows the tasker to create the environment on the fly using the VOV Environment Utilities.

This method offers several advantages: strict control on the environment, greater efficiency, less disk space utilization, easier execution across multiple platforms. This method is used if the environment variable `VOV_ENV` is defined; the value of the variable indicates the name of the environment to use.

 **Note:** This method is required for Windows.

Use Environment Snapshots

An environment snapshot will be created and used under the following conditions:

- The environment variable `VOV_ENV` is not set.
- The environment variable `VOV_ENV` is set to the value "" (the empty string) or the value `DEFAULT`.
- The environment variable `VOV_ENV` contains the substring `SNAPSHOT`.


The snapshot is represented by a file, the location of which is controlled by the environment variable `NC_SNAPSHOTDIR`. This variable can take one of the following symbolic values:

Possible values of <code>nc_snapshotdir</code>	
<code>homedir</code>	Use directory <code>~/ .ncsnapshots/\$VOVARCH</code>
<code>serverdir</code>	Use directory <code>PROJECTNAME .swd/snapshots/\$USER/\$VOVARCH</code>
any other value	Use the directory <code>\$LOGDIR/snapshots/\$USER/\$VOVARCH</code> , where <code>LOGDIR</code> is controlled by the variable <code>NC_LOGDIR</code> and has default value <code>./vnc_logs</code>

The environment snapshot is a file in Bourne-Shell syntax, which contains most of the variables in the current environment. The variables that are excluded from the snapshot include the following: `HOST OSREV OSTYPE TERMCAP SHELL PWD`. These variables are defined in the file `$VOVDIR/tcl/vtcl/vovenvutils.tcl`

An environment snapshot may be shared by many jobs.

When using a snapshot, the job is submitted with environment SNAPSHOT(name_of_snapshot_file).

 **Note:** This is a *named environment*.

To force the creation of an environment snapshot:

- Ensure sure the environment variable VOV_ENV is not defined.
- Do not use the option -e.

```
% unsetenv VOV_ENV
% nc run sleep 10
Resources= linux
Env       = SNAPSHOT(vnc_logs/snapshots/joe/linux/env4590.env)
Command  = vw sleep 10
Logfile   = vnc_logs/20020704/180936.7793
JobId     = 00350601
vnc: message: Scheduled jobs: 1          Total estimated time: 0s
```

Named Environments

The Accelerator Environment Utilities consist of two commands: `vel`, lists the available environments; `ves` switches between environments. For more information, refer to *Environment Management*.

The following example lists the available environments and switch to the environment called BASE.

```
% vel
vel: message: Environment directories:
1 /release/VOV/latest/sun5/local/environments
1 . tcl BASE          UNIX utilities, X windows, and VOV
1 . tcl D             Define vars: Usage: ves "+D(VAR1=value1,...)"
1 . tcl DEFAULT      Just a name for whatever you already have.
% ves BASE
```

Select a Named Environment

1. When submitting a job, to select the environment in which to run the job, use the option -e. Examples are shown below:

```
% nc run -e BASE sleep 10
...output omitted...
% nc run -e BASE+SPICE sleep 10
...output omitted...
% nc run -e "BASE+D(MYVAR=somevalue)" sleep 10
...output omitted...
```

Use Snapshot with Named Environment

1. A combination of an environment snapshot and a name environment can be set up. Try the following example shows using the -e option to set up a combined environment with a SNAPSHOT plus a name environment CALIBRE:

```
% nc run -e SNAPSHOT+CALIBRE sleep 10  
...output omitted...  
% nc run -e SNAPSHOT+MODULE1+CALIBRE sleep 10  
...output omitted...
```

Job Resources

Some jobs may require a specific platform or operating system (OS) to run. As Accelerator automatically lists platform/OS as a tasker resources, the task of finding the right host to execute a job is easy: specify the required platform/OS as part of the job resource requirements.

Cross-platform Job Runs

When submitting a job cross-platform, explicitly specify the environment for the job.

For example, the following command automatically sends the specified job to a Linux machine that uses the environment 'VERILOG'.

```
% nc run -e VERILOG -r linux -- verilog mydesign.v
```

Accelerator and ClearCase: nc run -clearcase

To submit a job in a ClearCase view, use the option `-clearcase` with `nc run`.

`CLEARCASE_ROOT` must be set. Otherwise, the option `-clearcase` has no effect.

```
% nc run -clearcase sleep 10
```

ClearCase presents two challenges for Accelerator when the taskers in the remote machines are not running in any special view:

- The working directory of the job, which could be within the `/vobtree`, is likely to not exist on the tasker machine. For the directory to exist, the view must first be initialized. In this scenario, `nc run -clearcase` submits all jobs from the HOME directory of the user. The actual working directory of the job is stored in the VOV Properties `VOVDIR` attached to the job. The view name is stored in the property `VIEW`, also attached to the job.
- The command needs to be executed in the view context, which is normally done with the following command:

```
cleartool setview -exec "COMMAND" VIEWNAME
```

As `COMMAND` can be complex, it is best to wrap it into an auxiliary script. `nc run -clearcase` is used as a wrapper for the command `vw ccexec`. The utility `ccexec` is executed on the remote tasker and performs the following steps:

- # Gets the view name from the property `VIEW` attached to the job
- # Gets the working directory of the job from the VOV Properties `VOVDIR`
- # Creates a temporary script in `/usr/tmp` to change directory and execute the command
- # Calls `clearcase setview -exec ... VIEWNAME`

Request a License Before Executing Jobs

Occasionally, a script can include a job that requires a license. Normally, such jobs are submitted to the queue.

However, instead of submitting the job to the queue, it can be checked if a license is available, and then execute the job on the local host. This can be automated by wrapping the command with `vovresreq`.

Use `vovresreq` with Accelerator

The `vovresreq` command works by waiting until the resource `License:hspice` is available before executing the remainder of the command line. In this example: the remainder of the command line is the SPICE run. If the license is not immediately available, `vovresreq` waits for the license before running the job.

Instead of calling `spice` directly, call it as follows:

```
% nc cmd vovresreq License:spice spice netlist.spi
```

`vovresreq`

This utility creates a job for tool `vovresgrab`.

```
vovresreq: Usage Message
```

DESCRIPTION:

This utility is normally used within scripts.

Execute a command after grabbing a list of resources (typically licenses) required to run the command. The utility waits until all requested resources have been made available.

The command also reads the `VNCSWD/vovresreq/config.tcl` file. This file contains the optional line:

```
set RESREQ(jobclass) <jobclassName>
set RESREQ(taskergroup) <taskergroupName>
set RESREQ(taskernames) <commaSeparatedListOfTaskerNames>
```

Currently the jobclass can be used but the resources it specifies are not currently used. But it is added here for a future release.

The taskergroup and taskernames are used in the resources of the `vovresreq` jobs. taskergroup if defined supercedes taskernames.

The jobclass name is the name of a jobclass to use for the resource check out. This jobclass usually specifies a tasker group to use that minimizes the number of job slots used. The taskers in this taskergroup are usually virtual taskers where the number of slots on the tasker is much larger than the number of cores. The `vovresreq` jobs are very minimal jobs and do not use a lot of cpu time and very little memory (much less than 20 meg). This job class should be sure to specify resources of

```
PERCENT/0 RAM/20
```

so the server can accommodate a large number of jobs. It is up to the job class creator to determine the actual value of the resources.

This utility creates a job for tool `vovresgrab`.

```
USAGE IN SCRIPTS: optional arguments are in []
```

```
Old style vovresreq command
  vovresreq [-v] RESOURCE_LIST [CMD]
in the above case the -v switch if defined must precede the
RESOURCE_LIST and CMD
This is exactly like submitting
  vovresreq [-v] -checkout RESOURCE_LIST -exec CMD
Both of these ways of using vovresreq are for single checkouts
and the checkout is done when the CMD is completed.
```

```
These ways of using vovresreq allow multiple checkouts and
check ins with the resid switch.
  vovresreq [-v] -checkout RESOURCE_LIST [-resid N] [-exec CMD]
  vovresreq [-v] -checkin  RESOURCE_LIST [-resid N]
```

```
USAGE WITH NetworkComputer:
% nc cmd vovresreq RESOURCE_LIST CMD ...
```

```
OPTIONS:
-h                -- display this usage message
-v                -- Increase verbosity.
-checkout        -- Wait for license to be checked out, then
                  return, else wait until the command completes.
-checkin         -- Return the resources previously checked out.
-exec            -- Execute command (this is the default)
-resid          -- Define the resource Id so they can be
                  checked in the order the user wants.
-sametasker     -- Make sure the vovresgrab runs on the same
                  tasker as the job currently executing
                  vovresreq. This is valuable for correct
                  matching of handles and jobs. However this can
                  be slow if the tasker is highly loaded.
```

EXAMPLES:

```
Example 1:
-- ...script fragment...
vovresreq -v "License:calibre" calibre cell.drc
```

```
Example 2:
-- ...script fragment for a single checkout and check in...
vovresreq -checkout "License:calibre" -resid 1
calibre cell.drc
vovresreq -checkin  "License:calibre" -resid 1
```

```
Example 3:
-- ...script fragment with multiple checkouts and check ins...
vovresreq [-v] -checkout RESOURCE_LIST1 [-resid 1]
calibre cell.drc
vovresreq [-v] -checkout RESOURCE_LIST2 [-resid 2]
calibre cell.erc
vovresreq [-v] -checkin  RESOURCE_LIST2 [-resid 2]
vovresreq [-v] -checkout RESOURCE_LIST3 [-resid 3]
calibre cell.lvs
vovresreq [-v] -checkin  RESOURCE_LIST3 [-resid 3]
vovresreq [-v] -checkin  RESOURCE_LIST1 [-resid 1]
```

```
Example 4:
-- Execute the command on the local machine but only after
-- grabbing a License:calibre resource.
% nc cmd vovresreq "License:calibre" -resid 1 calibre cell.drc
% nc cmd vovresreq "License:drc License:hsrc" -resid 2
calib bigcell.drc
```

vovresgrab

vovresgrab is used in conjunction with vovresreq to request resources on the fly. This utility is not meant to be invoked by the user, but only by vovresreq.

```
vovresgrab: Usage Message

DESCRIPTION:
    vovresgrab is used in conjunction with vovresreq
    to request resources on the fly.
    This utility is not meant to be invoked by the
    user, but only by vovresreq.

USAGE:
    % vovresgrab <RESLIST> <JOBID> <HOST> <PORT> <RESID>
```

NUMA Control and CPU Affinity

Non-Uniform Memory Access (NUMA) is to be used with machines that have multiple physical CPUs.

The performance of accessing RAM from a CPU depends on whether the RAM is physically attached to the same CPU or to another CPU. Therefore, application performance can be enhanced by constraining the application to stay within a single physical CPU. A Linux command, `numactl`, supports this control. For more information, use `man numactl`.

Accelerator can also automatically set the CPU affinity of an application in order to maximize its execution performance. The vovtasker automatically computes the socket, core, and memory layout of the machine on which it is executing. For each job that requests NUMA control, both the CORES and RAM resources for the job are used to determine where in the layout the job should be placed, depending on the placement type requested.

Note:

- If a job with NUMA control requested is executed on a machine with a single socket, NUMA will be ineffective, but no problems will occur.
- NUMA is supported only on Linux machines.

Placement Types

This version allows two different placement types for NUMA jobs: *pack* and *spread*. In both cases, the affinity of each single job is constrained to one or more physical sockets and have the following differences in behavior:

- Pack: the placement selects CPUs in a way that minimizes the number of unused cores in each CPU.
- Spread: the placement selects CPUs in a way that minimizes the loading of each CPU.

Examples of Job Submission with NUMA

```
% nc run -r CORES/2 -jpp pack -- my_job
% nc run -r CORES/4 -jpp spread -- my_job
```

To monitor the effect of CPU affinity, check the job property named "NUMA_AFFINITY":

For example:

```
% nc list -O "@ID@ @PROP.NUMA_AFFINITY@ @STATUSNC@"
283569646 NUMA pack: 0 * * * * * * * * * * * * * * * *
Running
283569650 NUMA pack: * 1 2 * * * * * * * * * * * * * * *
Running
283569654 NUMA pack: * * * * 4 5 6 7 * * * * 12 13 * *
Running
283569658 NUMA spread: * * * * * * * * * * * * * * 14 *
Running
283569662 NUMA spread: * * * * * * * * * * * * * * * 15
Running
283569665 NUMA spread: * * * 3 * * * * * * * * * * * *
Running
```

Check Tasker NUMA Status

Each tasker sets and maintains a property on itself named "NUMA_LAYOUT". The property is updated each time a job with a NUMA request begins or ends. To check the current NUMA status for a tasker named "foo":

```
% nc cmd vovselect prop.NUMA_LAYOUT from taskers where name==foo
Original:      Used/Total      _____+__
Socket: 0     RAM=      512/32089      ****oooooooooo
Socket: 1     RAM=     1024/32089     *****ooooo
```

CGROUPS for Jobs

A cgroup is a control group, used as a system for resource management on Linux.

A cgroup can be used to limit, throttle and account for resource usage per control group. Each resource interface is provided by a controller. Support for cgroup v2 is now enabled. For example, cgroups can be used to isolate core workloads from background resource needs. It prevents one workload from overpowering other workloads. On Linux taskers, a job can be requested to run in one or more cgroups.

cgroup v2

Below is a list of the fundamental differences between cgroup V1 and cgroup v2:

- Unified hierarchy - resources apply to cgroups now
- Granularity at TGID (PID), not TID level
- Focus on simplicity/clarity over ultimate flexibility

Other improvements are shown in the table below.

Description	v1	v2
Tracking on non-immediate/multi-source charges	No tracking of non-immediate charges Charged to root cgroup, essentially unlimited	Page cache writebacks and network are charged to the responsible cgroup


Description	v1	v2
		Can be considered as part of cgroup limits
Communication with backing subsystems	Most actions for non-share based resources reacted crudely to hitting thresholds For example, in the memory cgroup, the only option was to OOM kill or freeze	Many cgroup controllers negotiate with subsystems before real problems occur Subsystems can take remediative action (eg. direct reclaim withmemory.high) Easier to deal with temporary spikes in a resource's usage
Saner notifications	One clone() per event for cgroup release, expensive eventfd() support for others	inotify support everywhere eventfd() support still exists One process to monitor everything, if you like
Utility controllers make sense now	Utility controllers have their own hierarchies We usually want to use processes from another hierarchy As such, we end up manually synchronising	We have a single unified hierarchy, so no sync needed
Consistency between controllers	Inconsistencies in controller APIs Some controllers don't inherit values	Better consistency between controllers
Unified limits	Some limitations could not be fixed due to backwards compatibility memory. {,kmem.,kmem.tcp.,memsw., [...] }limit_in_bytes	Less iterative, more designed up front We now have universal thresholds (eg .memory.{high,max})

Syntax

The syntax is similar to requesting any other resource, with the resource name consisting of the prefix **CGROUP:** followed by the path to the cgroup on the filesystem relative to the root of the cgroup hierarchy.

In the following example, "sleep 120" job is assigned to the cgroup /cpuset/my_cgroup1 and /memory/my_cgroup2:

```
nc run -r CGROUP:/cpuset/my_cgroup1 -r CGROUP:/memory/my_cgroup2 -- sleep 120
```

 **Note:** The groups must exist and be constrained by the cgroup set up on the taskers. If that condition is not met, the job will not launch on the tasker because the resource must exist.

If multiple conflicting cgroups are assigned, such as two cgroups under the `/memory` hierarchy, the cgroup that is specified last is the cgroup that is assigned to the process.


The special resource `CGROUP:RAM` can be used to limit the memory usage of a job within a cgroup. In the following example, the job is assigned to a default cgroup that is limited to 2000 megabytes of RAM. As only one job is placed in each default cgroup, the RAM usage can be limited on a per job. The path to this default cgroup is:

```
<path to cgroup root directory>/memory/<queue name>_<tasker name>_<job slot number>
```

For example:

```
nc run -r CGROUP:RAM -r RAM/2000 -- sleep 120
```

There is a special case with RAM: specifying `CGROUP:RAM` and `RAM/200` would result in a job being placed in `/cgroup/memory/vncCG_buffalo_3` and `/cgroup/memory/vncCG_buffalo_3/memory.limit_in_bytes` being set to 209715200.

 **Note:** `CGROUP:RAM` cannot be used with a non-default cgroup. If both `CGROUP:RAM` and a non-default cgroup are specified, the job will be placed in the specified cgroup without changing that cgroup's RAM usage limit. We strongly recommend specifying a RAM resource when using `CGROUP:RAM`, as the default value is low (20 megabytes).

To see tasker resource for cgroups use, `nc hosts -r` and look for `CGROUP:` entries. If they are not present, ensure that they are set up on the taskers with `LSCGROUPS` and check the tasker logs for errors.

Enable cgroups

Many Centos, SLES or Ubuntu systems do not install with cgroups available by default. However, Use the steps below to configure Linux for cgroups.

On Centos 6, enable cgroup by installing the `libcgroup` RPM, and then enabling cgroup with the following:

```
% sudo service cgconfig start  
% sudo chkconfig cgconfig on
```

Use Containers for Jobs

Linux containers can be leveraged to constrain the amount of system resources used by jobs.

Containers are enabled by the administrator through named configurations that can be requested as a job resource. Each named configuration will contain a recipe of hooks to call to setup the container, and limits to enact upon them.

As a basic example:

```
% nc run -r Container:c1 -- sleep 120
```

The above job submission example will request that the tasker follow the recipe defined in the named configuration file for c1, to create the container described in the configuration and place the job into it.

The named container configuration may be defined to impose resource limits for the container being created. Some limits can be defined by the user through the use of resource requests in the job submission. There are three such requests that are considered during the processing of the named container configuration: CORES, RAM, and TMP.

CORES limit example:

```
% nc run -r Container:c1 CORES/2 -- sleep 120
```

RAM limit example:

```
% nc run -r Container:c1 RAM/20000 -- sleep 120
```

TMP limit example:

```
% nc run -r Container:c1 TMP#50000 -- sleep 120
```

RAM and TMP resources are specified in megabytes. Note that RAM is consumable, as indicated by the use of the / in the request. TMP, however, is not consumable, and such requests should use #. If a / is used with TMP, the amount of the TMP resource offered by the tasker will not be adjusted.

Match Jobs to Handles

Altair Accelerator tools work to match each license handle to the jobs currently running.

Typically, licenses are from FlexNet Publisher. This matching is based on "soft evidence"; the details of which process has checked out which handle are not available. Matching is based on the following information:

- The name of the user (normalized to lower case)
- The name of the host (normalized to lower case, possibly stripped of the domain name)
- The checkout time compared to the start of the job, with additional uncertainty as FlexNet Publisher reports checkout times to the minute, not to the second.

Matching is especially challenging when jobs of similar age are running on the same machine by the same user.

Despite these difficulties, the matching routines produce useful information that can be used to learn which licenses are actually used by a job. This information is also used by the preemption module.

For information about matching, check the following fields:

lmhandlesall	A list of all matched resources, consisting of one or more triplets of the form resourcename: tokens match_type handle_id"
lmhandlesnru	A list of only the not requested but used (NRU) resources
lmresources	A list of all matched resources, in a older format; use lmhandlesall instead

Summary of Matching Status Values

Legend for Matching Status	
sure	This is a Sure match. There is only one handle that matches the job.
best	This is the Best we can do .
also	This could also be a match, but not certain. The number of also matches can be limited by setting the parameter <code>resusermaxmatches</code> . The default value is 6.
group	This is a match if Grouped with another handle.
Old	This is a match with an Old job : a job that completed recently.
nru	Not-Requested/Used : Not a match. This handle matches a job that does not request the resource.
rnu	Requested/NotUsed : Not a match. This job does not use the resource it requested.
revoked	Revoked.
part revoc	Partial Revocation . This can occur when preemption <i>steals</i> only some of the tokens for a feature, not all. Typically, this is considered an error.
too early	Too early to tell . This applies to jobs that are younger than one minute.
ooq	Not a match . This appears to be a handle used outside of the queues .


For additional information about handle matching, and when disabling matching jobs could be needed, refer to [Resource/Handle Matching](#).

Resource/Handle 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.

 **Note:** 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, we recommend disabling the handle matching for selected licenses.

```
#
# Example of disabling matching for a couple of resources for which
# matching may be too expensive.
# This is to be done in the file resources.tcl
# (or vovresourced/config.tcl)
#
vovresSetFlags License:VCSRruntime_Net -nomatch -noooq -norecent
vovresSetFlags License:NC-Verilog      -nomatch -noooq -norecent

### The default status for the resource flags can be recovered as
### follows:
# vovresSetFlags License:abc -match -ooq -recent -log -nooverbook
```

Control Matching

The matching can be disabled for a resource by setting the `-nomatch` option in `vtk_resourceemap_set` for the resource.

```
# Example:
vtk_resourceemap_set License:abc -max 100 -lmfeature abc -nomatch
```

There are also a few global parameters (that is, the parameters set in `policy.tcl`) that control the matching:

- `resusermatchtolerance`** In seconds, determines a tolerance in matching checkout timestamps with jobs starts
- `resusermaxmatches`** The number of "also" matches that we look for. .
- `resuserDisableMatchingThreshold`** A threshold for disabling matching if the sum of Monitor handles and FlowTracer jobs exceeds it.

Advanced Information

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.

Node Fields

Each node in the trace has *field* attributes.

There are three types of fields:

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

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

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

```
% vovshow -fields
```

Incompatible Fields

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

Fields List

Field Name	Field Type	Applies To	Description
AGE	Integer (duration)	Nodes	Age in seconds of a job or of a file.
AGEPP	String	Nodes	Pretty-printed version of age of a job or of a file.
ANNOTATIONS	Integer	Nodes	Number of annotations attached to the node.
AUTOKILL	Integer (duration)	Jobs	Time before a job is automatically killed.
AUXRESOURCES	String	Jobs	Additional resources assigned to a job by the server.
BUCKETID	String	Jobs	The internal ID of the bucket object that contains the specified job.
CHOSENTASKERID	String	Jobs	If a job must go to a specific tasker, the ID of that tasker.
COMMAND	String	Jobs	The command line for the job.
COMMANDLENGTH	Integer (size)	Jobs	The length of the command line for the job.
CPUPROGRESS	Integer (percent)	Jobs	The percent of CPU time used by a jobs, including all its children, in the last sampling interval. This number can be greater than 100 if the job is running on a multi CPU machine.

Field Name	Field Type	Applies To	Description
			If CPUPROGRESS is zero, the job is stuck, probably waiting for input or for a license.
CPUTIME	Integer (milliseconds)	Jobs	The CPU time used by the job, in milliseconds, including all its children. This is a 64-bit number.
CURRAM	Integer (MB)	Jobs	The current amount of RAM used by the job, in MB
CURREAD	Integer (bytes)	Jobs	The current number of input I/O events for the job, in bytes.
CURVM	Integer	Jobs	The current amount of virtual memory used by the job, in MB
CURWRITER	Integer (bytes)	Jobs	The current number of output I/O events for the job, in bytes.
CWD	String	Jobs	The current working directory for the job.
DB	String	Files	The database of a file.
DIR	String	Nodes	Same as CWD.
DISPATCHDATE	Integer (timestamp)	Jobs	The time the job has been dispatched to a tasker. This is relevant for indirect taskers, where the DISPATCHDATE may be significantly different from the START time due to latency in the secondary queue.
DURATION	Integer (duration)	Jobs	The duration of a job, in seconds.
DURATIONPP	String	Jobs	The duration of a job ID in pretty format.
ENDDATE	String	Jobs	The end date of a job in string format.
ENDED	Integer (timestamp)	Jobs	The end date of a job in integer format. If the job is still running or retracing, this field has the value 0 (see Ended2 for a different value).
ENDED2	Integer (timestamp)	Jobs	The end date of a job in integer format. If the job is running or retracing, this field returns the current time stamp (see ENDED for a different value)
ENV	String	Jobs	The environment of a job.
ENVARGS	String	Jobs	From the environment string, all the words after the first.
EXECHOST	String	Jobs	The execution host of the job.
EXITSTATUS	Integer	Jobs	The exit status of a job.

Field Name	Field Type	Applies To	Description
EXPDUR	Integer (duration)	Jobs	(OBSOLETE! Use XDUR instead) Expected duration of a job, in seconds
EXPDURPP	String	Jobs	(OBSOLETE! Use XDURPP instead) Expected duration, pretty printed.
FAILCODE	Integer	Jobs	A mask of values indicating why a job failed.
FLAGS	String	Nodes	Obsolete. Not supported any longer.
FSEXCESS	Integer	Jobs	This is essentially (FSHISTORY - FSTARGET) + (FSRUNNING - FSTARGET). See also FSEXCESSRUNNING and FSEXCESSHISTORY
FSEXCESSHISTORY	Integer	Jobs	This is essentially (FSHISTORY - FSTARGET). See also FSEXCESS.
FSEXCESSRUNNING	Integer	Jobs	This is essentially (FSRUNNING - FSTARGET). See also FSEXCESS. If the number is positive, that means that the group to which the job belongs has more running jobs than it should.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	VovPreemptRule -rulename SOMENAME \ ... -preempting "FSEXCESSRUNNINGLOCAL<0" \ -preemptable "FSEXCESSRUNNINGLOCAL>0 JOBCLASS==@JOBCLASS@ FSRANK9>@FSRANK9@" \ ...
FSEXCESSRUNNINGLOCAL	Integer	Jobs	Similar to FSEXCESSRUNNINGLOCAL but uses the balance of running jobs at one level above the local.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	Similar to FSEXCESSRUNNINGLOCAL but uses the balance of running jobs at two levels above the local.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	Similar to FSEXCESSRUNNINGLOCAL but uses the balance of running jobs at three levels above the local.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL1.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL2.
FSEXCESSRUNNINGLOCAL	Integer	Jobs	This is number of jobs corresponding to FSEXCESSRUNNINGLOCAL3.

Field Name	Field Type	Applies To	Description
FSGROUP	Integer	Jobs	The FairShare group that the job belongs to.
FSHISTORY	Integer	Jobs	The actual share in the FairShare window for the FairShare group, multiplied by 10,000.
FSRANK	Integer	Jobs	The FairShare rank of the FairShare group to which the job belongs, or -1 if the FairShare group has no rank. (See also FSRANK9)
FSRANK9	Integer	Jobs	This is the same as FSRANK, except that the value returned for groups that have no FairShare rank is 9,999,999 instead of -1.
FSRUNNING	Integer	Jobs	The actual share of all running jobs for the FairShare group, multiplied by 10,000.
FSRUNNINGCOUNT	Integer	Jobs	The current number of running jobs for the FairShare group of the job.
FSSUBGROUP	String	Jobs	The part of the FairShare group after the colon.
FSSUSPENDED	Integer	Jobs	The current number of suspended jobs for the FairShare group of the job.
FSTARGET	Integer	Jobs	The FairShare target for the FairShare group to which the job belongs. The target, which is normally a fractional number less than 1.0, is multiplied by 10,000 to yield the FSTARGET. For example, a group that has a target of 30%=0.3 will have a FSTARGET of 3,000.
FSTOKENS	Integer	Jobs	An integer multiplier for the contribution of the job to the FairShare (e.g. a fstokens value of 2 means that the job counts as 2 "normal" jobs)
FSUSER	String	Jobs	The user component of a FairShare node (i.e. the component right after the dot '.', if it exists)
GRABBEDRESOURCES	String	Jobs	The list of resource maps that have been grabbed in order to dispatch a job to a tasker. This is valid only for RUNNING and RETRACING jobs, and the value of the field may change over time due to reconciliation of resources. See also the GRABBEDRESOURCESO field, which is available for jobs even after completion.
GRABBEDRESOURCESO	String	Jobs	The original list of grabbed resources when a job was last dispatched to a tasker. For RUNNING and RETRACING jobs it is best to look at GRABBEDRESOURCES instead. The final letter is an "oh" for Original.

Field Name	Field Type	Applies To	Description
GRABBEDTOOLS	String	Jobs	Obsolete: Only for running jobs
GROUP	String	Jobs	The FairShare group of a job. This is different from OSGROUP, which is used to specify the group permissions for the job.
HASANNOTATIONS	Boolean	NODES	True if the node has annotations.
HASINPUTCONFLICT	Boolean	JOBS	True if job failed on an input conflict.
HASINPUTS	Boolean	NODES	True if a node has one or more inputs.
HASOUTPUTCONFLICT	Boolean	JOBS	True if job failed on an output .conflict
HASOUTPUTS	Boolean	NODES	True if a node has one or more outputs.
HASPROFILE	Boolean	JOBS	True if job collects profile information like RAM and CPU usage at runtime (see option -profile in nc run)
HASRUNINFO	Boolean	JOBS	Obsolete: always 0.
HOST	String	JOBS	The host that executed a job.
ID	String	NODES	The ID of a node. This field is of type "string" and is shown with leading zeroes (e.g. "000123456"). Contrast this with the field IDINT which is of type "integer".
IDINT	Integer	NODES	The integer version of the Vovid of a node. Contrast this with the field ID which is of type "string"
INPUTS	Integer	NODES	The number of inputs of a node.
ISAUTOFLOW	Boolean	JOBS	The job turns VALID as soon as all its inputs are VALID, with no execution.
ISAUTOFORGET	Boolean	JOBS	The job is automatically forgotten after a certain time after completion.
ISAUTOKILL	Boolean	JOBS	The job is automatically killed if it exceeds its expected duration.
ISBARRIER	Boolean	NODES	True if a file has a barriers on it.
ISBARRIERINVALID	Boolean	NODES	True if a file has a barriers would be INVALID were it not for the presence of a barrier.
ISCHAIN	Boolean	FILES	True if the file is part of a chain, meaning that there are tools that operate 'in-place' on the file.

Field Name	Field Type	Applies To	Description
ISDATA	Boolean	NODES	True if node is a file.
ISEPHEMERAL	Boolean	JOBS	(experimental, do not use).
ISFILE	Boolean	NODES	True if node is a file.
ISINTERACTIVE	String	JOBS	True if the job is interactive (-I -Ir -Il -wl)
ISJOB	Boolean	NODES	True if node is a job.
ISMIGRATABLE	Boolean	JOBS	This field is an annotation to identify jobs that can be migrated. Currently not supported.
ISNODE	Boolean	NODES	True if a node is a node (Always true)
ISNONEXEC	Boolean	JOBS	True if a job is not executable
ISNONEXEC	Boolean	JOBS	True if the job is non-executable.
ISOPTIONAL	Boolean	FILES	True if an output has the OPTIONAL flag
ISPLACE	Boolean	NODES	OBSOLETE: True if a node is a file (see ISFILE)
ISPREEMPTABLE	Boolean	JOBS	This field is honored by the <code>vovpreemptd</code> daemon. If the value is zero, then the job is not considered for preemption. For information on how to set the preemptable flag, see <i>Control Whether a Job is Preemptable</i> . See also ISMIGRATABLE.
ISPROCESS	Boolean	NODES	Same as ISJOB.
ISQUEUED	Boolean	JOBS	The job is ready to fire and in a bucket in the job queue.
ISREADYTOFIRE	Boolean	JOBS	All inputs of a job are VALID and the job is ready to fire.
ISSCHEDULED	Boolean	NODES	True if node is scheduled to be retraced.
ISSCHEDULEDBARRIERINV	Boolean	NODES	True if node is scheduled and had the barrier invalid flag set.
ISSHARED	Boolean	FILES	True if file is shared output.
ISSKIP	Boolean	JOBS	True if the job is skipped (also called ISAUTOFLOW).
ISSTDERR	Boolean	FILES	True if file is a stderr file.
ISSTDOUT	Boolean	FILES	True if file is a stdout file.
ISSUBJOB	Boolean	JOBS	True if the job is a subjob (e.g. partialtool, vovjobresumer).
ISSUSPENDED	Boolean	JOBS	True if the job or one of its children is suspended.

Field Name	Field Type	Applies To	Description
ISSYSTEMJOB	Boolean	JOBS	True if the job is a 'system job' like vovzip, or vovsh -s netinfo.
ISTOOL	Boolean	NODES	Same as ISJOB and ISPROCESS.
ISTRANSFER	Boolean	JOBS	True if the job is being transfered to another cluster.
ISTRANSITION	Boolean	NODES	True if node is job (see ISJOB)
ISTRIGGERRUN	Boolean	FILES	True if the file has the trigger,run flag, i.e. if a change in the file triggers a retrace of the downcone
ISTRIGGERSTOP	Boolean	FILES	True if the file has the trigger,stop flag, i.e. if a change in the file triggers a stop of the downcone, followed by a retrace of the downcone
ISUNSAFE	Boolean	JOBS	True if node is "unsafe"
ISZIPPABLE	Boolean	PLACES	True if the file can be automatically zipped.
ISZIPPED	Boolean	PLACES	True if the file is currently zipped.
IUO	String	NODES	One of the following characters: "." "i" "u" "o"
JOBCLASS	String	JOBS	The jobclass of a job
JOBID	String	JOBS	The ID (number) of a job (a field of PROCESSES). An integer but often with leading 0s.
JOBLOGDIR	String	JOBS	The directory to which the logfile of a job is written (if a logfile is specified on the command line). If no logfile is specified, this is the current directory of the job.
JOBNAME	String	JOBS	The job name of a job
JOBPROJ	String	JOBS	Name of the project that the job belongs to (same as project).
JPP	String	JOBS	Job placement policy.
LABEL	String	NODES	A short label for the node.
LASTCPUPROGRESS	Integer (timestamp)	JOBS	The last time the system detected some cpu progress in the job.
LASTCPUPROGRESSPP	String	JOBS	The pretty-print version of "\$NOW-LASTCPUPROGRESS".
LEGALEXIT	String	JOBS	The legal exit allowed for a job to be considered VALID (same as OKSTATUS).
LEVEL	Integer	NODES	The level of a node.

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

Field Name	Field Type	Applies To	Description
OKSTATUS	String	JOBS	The list of acceptable exit status (same as LEGALEXIT).
OSGROUP	String	JOBS	The operating system group for the job (different from GROUP, which is used in the FairShare scheduling).
OUTPUTS	Integer	NODES	The number of outputs of a node.
PID	Integer	JOBS	The process id of a job.
PRIORITY	Integer	JOBS	Prioity of the job, in the range 0 to 15.
PRIORITYPP	String	JOBS	Pretty printed version of the priority of the job.
PROJECT	String	JOBS	Name of the project that the job belongs to (same as jobproj)
PROP.<propname>	String	NODES	This field is used to access the property with name <code>propname</code> . For example, to access the property "ABC", one should ask for "PROP.ABC". See also field PROPERTIES.
PROPERTIES	String	NODES	The list of properties attached to a node. The list consists of an even number of words, where the first word is either 'S' (for STRING) or 'I' for INTEGER properties and the second word is the name of the property. See also the 'PROP.*' field
QUEUETIME	Integer (timestamp)	JOBS	Time the job was put onto the job queue, more specifically the last time the job was scheduled. The job could in fact enter the job queue later due to dependency constraints or to lack of space in queue.
QUEUEWAIT	Integer (duration)	JOBS	The time a job has waited in the queue, in seconds.
QUEUEWAITPP	String	JOBS	Pretty-print version of QUEUEWAIT.
RANDOM	Integer	ALL	A random number in the range [0-9999]. Used, for example, to sort jobs in preemption rules.
REQCORES	Integer	JOBS	Requested CORES for the job. This is set when the job is dispatched to a tasker. Another name for this field is REQCPUS. See also explanation on REQRAM.
REQPERCENT	Integer	JOBS	Requested PERCENT for the job. See also explanation on REQRAM.
REQRAM	Integer (MB)	JOBS	RAM requested by a job. This value may come from 1) the SOLUTION property on the job, if it exists, 2) from the requested resource on the bucket, if the job is queued, or 2) from the resources string of the job. Normally, the value is the

Field Name	Field Type	Applies To	Description
			same regardless of the origin, but it is possible and acceptable that the value may change due to additional math performed during scheduling. Normal case: a job requests "RAM/200". Then the REQRAM value is going to be 200. Strange case: a job requests "RAM/200 SLOTS/2 RAM/300" (i.e. the RAM request is repeated). In this case, REQRAM will be 200 for the job if it is not Queued or if the SOLUTION property is removed after the execution of the job. Else it will be 500 (=200+300) while the job is Queued.
REQSLOTS	Integer	JOBS	Requested number of slots for the job. See also explanation on REQRAM.
REQSWAP	Integer (MB)	JOBS	Requested swap for the job. See also explanation on REQRAM.
RESERVEDRESOURCES	String	JOBS	DO NOT USE: Only for running jobs
RESERVEDTOOLS	String	JOBS	DO NOT USE: Only for running jobs
RESOURCES	String	JOBS	The resources of a job.
RETRACINGID	String	JOBS	The id of the retracing job (to find the other job is a RUNNING/RETRACING pair).
RUNSTATUS	String	JOBS	A representation of how well the job is running. Typical values are Good, Paging, NoCpu. Check <i>Job Runtime - Monitor and Profile</i> for more information.
SCHEDTIME	Integer (timestamp)	JOBS	The earliest time that this job can be scheduled (set with -at or -after option in nc run).
TASKERGROUP	String	JOBS	The tasker group of the tasker to which the job has been dispatched.
TASKERID	String	JOBS	The id of the tasker to which the job has been dispatched.
TASKERNAME	String	JOBS	The tasker name to which the job has been dispatched.
TASKERSLOTSSUSPENDAB	Integer	JOBS	Number of suspendable jobs on the tasker on which the job is running.
TASKERSLOTSSUSPENDED	Integer	JOBS	Number of slots suspended on the tasker on which the job is running.
TASKERSLOTSUSED	Integer	JOBS	Number of slots used on the tasker on which the job is running.
STARTDATE	String	JOBS	Like STARTED, only in nice formatted ASCII

Field Name	Field Type	Applies To	Description
STARTED	Integer (timestamp)	JOBS	The time the job was started, or -1 if the job never ran.
STATUS	String	NODES	The status of the node.
STATUSCOLOR	String	NODES	The color associated to the node
STATUSCONSTRAST	String	NODES	A color that contrasts with STATUSCOLOR. Used mostly in templates for the browser interface.
STATUSIO	String	NODES	Short version of STATUS. For example, instead of VALID you get 'V' (the name is not very good)
STATUSMASK	Integer	NODES	(HARD TO USE) a binary mask with a bit set for each value of node status.
STATUSNC	String	NODES	job status for Accelerator.
STOLENRESOURCES	String	JOBS	These are the resources that have been taken away (revoked) from the job, typically because the job has been preempted.
SUBJOBIDS	String	JOBS	The ids of all subjobs of a job (used for example with distributed parallel jobs).
SUBMITHOST	String	JOBS	The submission host
SUBRESOURCES	String	JOBS	Subordinate resources for a job, i.e. the resources after the first '--' token. Used by indirect taskers.
SUSPENDEDINTERVALS	String	JOBS	The list of all intervals in which the job has been suspended.
SUSPENDEDTIME	Integer (duration)	JOBS	The total amount of time the job has been suspended (in seconds).
SUSPENSION	Integer (duration)	JOBS	The same as suspendtime.
TAIL	String	FILES	The name of the file not including the directory path.
TIMESTAMP	Integer (timestamp)	FILES	The time the file was last modified.
TIMESTAMPPPP	String	FILES	Pretty-printed version of TIMESTAMP.
TOOL	String	JOBS	The name of the first tool used by the job.
TOPJOBID	String	JOBS	The id of the top job in a distributed parallel job.
USER	String	JOBS	The user (or owner) of a job.

Field Name	Field Type	Applies To	Description
USERXDUR	Integer (duration)	JOBS	The expected duration of a job as specified by the user. If negative, the expected duration has not been specified.
USERXDURPP	String	JOBS	The expected duration as specified by the user, pretty printed.
USESESSIONID	Boolean	JOBS	True if the vovtasker is supposed to look at the session id to find the processes related to this job (in addition to the parent-child relationship).
WAITREASON		JOBS	A description of why this job is waiting.
X	Integer	NODES	The X coordinate of a node.
XDUR	Integer (duration)	JOBS	The current expected duration of a job. It may be set by the user or calculated from the most recent successful completion of the job. If negative, the expected duration is not known.
XDURPP	String	JOBS	The current expected duration of a job, pretty printed.
XPRIORITY	Integer	JOBS	The execution priority of a job (range 0 to 15)
XPRIORITYPP	String	JOBS	Pretty-printed version of XPRIORITY.
Y	Integer	NODES	The Y coordinate of a node (same as LEVEL).
YOUNG	Boolean	FILES	This field is 1 if the file is younger than any of the jobs that need it. Young nodes are shown with a lighter shade of green in the GUI.
Z	Integer	NODES	The Z coordinate of a node (always 0).
ZIPPED	String	FILES	The string 'ZIP' if the file is zipped, or the empty string.

Formatting Strings

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

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

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

Field Reference

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

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

Size Specification

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

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

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

For example:

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

Time Specifications

Some VOV procedures and commands accept as input a time specification, which is a string that contains a mixture of digits and the letters **s m h d w**.

The letters are defined as follows:

Specification	Explanation
a	Seconds (default)
m	Minutes
h	Hours
d	Days (24 hours)
w	Weeks (168=7*24 hours)

The time specifications are case insensitive. The **d** and **w** specifications ignore that with daylight-saving some days may be 23 hours and other days may be 25 hours.

Examples of TimeSpecs

Specification	Explanation
60	60 seconds
2M	2 minutes, i.e. 120 seconds
3h30m	3 hours and 30 minutes, i.e. 12600 seconds

You can convert a time specification to seconds with the Tcl procedure `VovParseTimeSpec`. Conversely, you can convert an integer to a time specification, but with some loss of precision, with the procedure `vtk_time_pp`.

Some utilities (such as `ftlm_batch_report`) require a time interval specification. Time intervals can be expressed as follows:

- past hour
- today
- yesterday
- this week
- last week
- past week
- this month
- this month full
- last month
- past month
- past 30days
- this quarter
- last quarter
- this year
- last year
- YYYY, such as 2016, would be the entire year of 2016
- YYYYMM, such as 201016, which would be the month of December in 2016

- YYYYMMDD, such as 20100116, which would be Jan 15 2016
- YYYYwWW, such as 2017w4, which would be week 4 in year 2017
- Month YYYY. Example: Sep 2017
- start-finish, where on each side of the '-' is a timestamp specification that is parsed by VovScanClock. Example: 20090101-20090301

The conversion to a start-end pair is performed by the Tcl procedure `VovDate::computeSymbolicInterval`

Selection Rules

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

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

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

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

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

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

is evaluated as:

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

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


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

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

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

This rule is equivalent to:

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

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

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


Examples of Selection Rules

Selection Rule	Explanation
<code>isjob==1</code>	Select all jobs
<code>isjob</code>	Select all jobs (equivalent to <code>"isjob!=0"</code>)
<code>!isjob</code>	Select everything except jobs (equivalent to <code>"isjob!=0"</code>)
<code>not isjob</code>	Select everything except jobs (equivalent to <code>"isjob!=0"</code>)
<code>IS_FILE and db!=FILE</code>	Select all files which are not in the database FILE
<code>status==RUNNING</code>	Select all nodes whose status is RUNNING
<code>status==RUNNING,VALID</code>	Select all nodes whose status is RUNNING or VALID
<code>status!=RUNNING,VALID</code>	Select all nodes whose status is neither RUNNING nor VALID
<code>status!=RUNNING and status!=VALID</code>	Select all nodes whose status is neither RUNNING nor VALID
<code>isjob status==INVALID</code>	Select all invalid jobs
<code>isjob duration>300</code>	Select all jobs that have lasted more than 5 minutes (i.e. 300 seconds).
<code>isjob command~~aa</code>	Select all jobs with a command line containing the string aa; the ~~ operator is used for case insensitive match.
<code>isjob age<600</code>	Select all jobs completed less than 10 minutes ago.
<code>isjob & age<600</code>	Select all jobs completed less than 10 minutes ago.
<code>isjob AND age<600</code>	Select all jobs completed less than 10 minutes ago.
<code>isfile name~ccc</code>	Select all files with name containing ccc.
<code>isjob inputs<2 status==INVALID</code>	Select all jobs that have fewer than 2 inputs and are invalid.
<code>isjob AND (inputs<2 OR status==INVALID)</code>	Select all jobs that have fewer than 2 inputs or are invalid.

Selection Rule	Explanation
<code>isjob (inputs<2 status==INVALID)</code>	Select all jobs that have fewer than 2 inputs or are invalid.
<code>isjob inputs<2 status==INVALID</code>	Select all jobs that have fewer than 2 inputs, and all nodes (including non-jobs) that are invalid. See notes on AND/OR precedence, above.
<code>isfile status==VALID age>=600 name~xxx</code>	Select all valid files older than 10 minutes whose name contains the string xxx.
<code>isfile status==VALID age>=10m name~xxx</code>	Select all valid files older than 10 minutes whose name contains the string xxx.
<code>isjob tool==gcc resources~diskio duration>10</code>	Select all jobs that use the tool gcc and require the resource diskio and take more than 10 seconds.
<code>isjob status~A,D</code>	Select all jobs that have either "A" or "D" in their Status fields; see notes on commas in regular expressions, above.
<code>isjob & status~"A,D"</code>	Select all jobs that have the string "A,D" in their Status fields.
<code>isjob & status~[A-Z]+A[A-Z]+D</code>	Select all jobs that match the regular expression "[A-Z]+A[A-Z]+D" in their Status fields (e.g. FAILED, INVALID).
<code>isjob status::inv*</code>	Select all jobs whose status fields start with "inv" (case-insensitive).
<code>isjob and status:inv*</code>	Select all jobs whose status fields start with "inv" (case-sensitive).
<code>isjob status!::inv*</code>	Select all jobs whose status fields do not start with "inv" (case-insensitive).
<code>isjob AND command!^"sleep 60"</code>	Select all jobs whose commands do not contain the string "sleep 60" (case-sensitive). Note that spaces are allowed in quoted values.
<code>isjob AND command!^^sleep\ 60</code>	Select all jobs whose commands do not contain the string "sleep 60" (case-insensitive). Note that spaces are allowed if preceded by a backslash ("\").

Selection Rules and Incompatible Fields

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

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

For example:

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

Migrate from LSF

If you are a user of LSF™ moving to Accelerator, you need to be aware of some terminology shift.

For example, in Accelerator terminology, the word 'queue' has a different meaning from the one used by other batch systems. For Accelerator, a queue is a group of compute hosts managed by a single Altair Accelerator vovserver program. The name of the queue should start with the letters 'vnc'.

The following table summarizes the differences in terminology between LSF and Accelerator.

Table 2: Comparison of Main Concepts

LSF	Accelerator
LSF master	vovserver
LSF (compute) server	vovtasker
LSF cluster	VOV queue or VOV project
LSF queue	VOV jobclass

Table 3: Comparison of Main Commands

LSF	Accelerator
bsub	nc run
bjobs	nc list,nc monitor,nc gui
bstop	nc stop
lshosts	nc hosts

To ease the migration from LSF to Accelerator, a set of [emulation scripts](#) are provided for commands such as `bsub` and `bjobs`. These scripts can be found in `$VOVDIR/scripts/lsfemulation`.

Emulating LSF_JOBINDEX

In Accelerator, job arrays are just a simple way to submit lots of jobs that are pretty similar. Each element of the array is actually a regular job. You can pass the index of the job in the array by variable substitution. To emulate LSF_JOBINDEX, you can use the following technique:

```
% nc run -e SNAPSHOT+D,LSF_JOBINDEX=@INDEX@ -array 100 someCommandWithArgs
```

Frequently Asked Questions and Troubleshooting Tips

I'm doing an installation and configuration in a Windows environment - can I use PowerShell?

PowerShell is not supported; we strongly recommend not using PowerShell.

How do I contact Altair Engineering to get additional support, report a bug, or request a feature?

You can contact Altair Engineering at: <https://www.pbsworks.com/ContactSupport.aspx>.

Why can't I access Monitor's historical license usage through Accelerator?

Accelerator ships with a version of Monitor that is licensed to monitor current license activity only. This edition is called LMS (Monitor Small). To access Monitor's historical license usage information, you must have the full version of Monitor.

What do I do in the event of a server failover or crash?

You can find a checklist for system recovery on the System Recovery page. You can find this address with the command:

```
nc cmd vovbrowser -url /cgi/sysrecovery.cgi
```

Where is the policy.tcl file? What about the taskers.tcl file? The resources.tcl and security.tcl files?

All .tcl configuration files for Accelerator are located at \$VOVDIR/./././vnc/vnc.swd

How do I enable the retrace of more than 400 jobs at a time?

The limit to how many jobs can be run/retraced at any given moment is defined by the *maxNormalClients* config variable. To change the variable, you can use the command:

```
vtk_server_config "maxNormalClients" maxnumberofjobs
```

How do I receive email notifications on job completion?

To receive automatic notification of major FlowTracer and Accelerator events, you should use the *vovnotifyd* daemon.

How do I track the memory usage of taskers?

VOV automatically keeps track of tasker memory usage. *vovtasker* keeps logs of 1 minute, 5 minute, and 10 minute load averages of the machines where taskers are running on. The tasker reports are available on the Tasker Load page. The Accelerator URL can be found with the command:

```
nc cmd vovbrowser -url /cgi/taskerload.cgi
```

Why are my jobs taking so long?

There are multiple reasons why FlowTracer jobs may be retracing slowly. Fortunately, Accelerator produces reports to help diagnose any problems. Read about available reports at *Resource Plots* in the Altair Accelerator User Guide..

How do I change to another version of Accelerator?

To upgrade Accelerator software, refer to [Upgrade Accelerator](#).

How do I access information on license usage?

Accelerator does not have this functionality. This functionality belongs to Monitor. If you have Monitor installed and fully working, you can display license information at on the FTLM page.

Why are my licenses not fully utilized? I'm sure they're completely booked.

Your licenses are not fully utilized probably because they are not [overbooked](#).

Essentially, the jobs being run do not use a license 100% of the time. Because there are jobs booked for licenses 100% of the time, there will be times where licenses are not utilized. This is because one or more jobs will still be running, but be done with the license that was booked. To rectify this, jobs are queued for more than 100% of the licenses, allowing another job to start and utilize the open license.

How do I share licenses between jobs in queue?

Read more on license at [License Sharing Support](#).

My tasker is sick! What do I do?

Your tasker is sick because it has not sent out a heartbeat for at least 3 minutes. This may mean your tasker has crashed or disconnected. Once you have identified a sick tasker, you can proceed to troubleshoot it to fix the problem.

This list may be helpful:

- Check to make sure the machine itself is healthy. Make sure it is running, connected to the network, and not jammed.
- Check to see if vovtasker or vovtaskerroot is still running. If it isn't, then the tasker program itself has crashed. You should restart the tasker program with:

```
vovtaskermgr start
```

- Check to see if vovtasker or vovtaskerroot is stuck. If it is, Linux commands such as `strace` and `pstack` should provide you with enough information to fix it.

My tasker is healthy, but all jobs sent to the tasker come out failed. What is going on?

Your tasker is what is called a black hole. It appears healthy, but is in fact unable to execute jobs. There is functionality to enable automatic detection of black holes in the [Black Hole Detection](#) page.

When a black hole is found, it would be prudent to send a simple job such as `cp` or `sleep` to the tasker to confirm its black hole state.

I want to give a different amounts of resources to different sites. How can I do that?

[FairShare](#) is a mechanism to allocate CPU cycles among groups and user according to a policy. This would be your best bet.

How do I limit a resource for a particular user?

Although it is not recommended, information on limiting users can be found on the [Limit Users](#) page.

My job was killed because it failed to start within 1m00s!

This can be caused by a bad NFS mount point, or an automounter that is so overloaded, that it fails to mount the run directory for the job in under a minute. Although this is a hardware problem, there is a workaround by changing the variable `VOV_MAX_WAIT_NO_START` to a value over 1 minute.

How do I setup prioritized licence usage?

For example, to use the licence FOO_BAR_A first, then the licence FOO_BAR_B second, use:

```
vtk_resourcemap_set FB-lic UNLIMITED "Licence:FOO_BAR_A OR Licence:FOO_BAR_B"
```

In the jobclass. To set it in a resource map, use:

```
set VOV_JOB_DESC(resources) "Licence:FOO_BAR_A OR Licence:FOO_BAR_B"
```

Why am I missing the plots when I look at a resource or license report?

Probably, what is causing the plots to be missing is a name resolution issue. To fix this, make sure VOV_HOST_HTTP_NAME is set correctly. If all else fails, set this to the host's IP address, not network name. To update a running server, you must use the command:

```
vtk_server_setenv VOV_HOST_HTTP_NAME XXX
```

vovresourced is not starting, says 'Failed to source' too many resources'!

Most likely, you have exceeded the limit for resource maps in use. To raise this limit, change the maxResMap value in policy.tcl.

How do I ensure that a tool is preemptable robustly?

Sometimes a tool will crash when preempted. To test whether this is Altair, or the tool vendor, try and run the tool without Altair binaries (pure UNIX code) and see if the tool still crashes. The steps to do this are as follows:

1. Start the EDA tool(s) which you wish to test.
2. Use the UNIX command ps to find the PID of the EDA tool(s):

```
% ps | grep firefox
PID TTY          TIME CMD
349 ?            00:24:19 firefox
```

3. Send TSTP and CONT signals 10 seconds apart repeatedly. Try this in your shell:

```
% kill -TSTP 349 ; sleep 10 ; kill -CONT 349 ; sleep 10 ; kill -TSTP 349 ;
(etc...)
```

Following these steps, if the tool crashes, then the problem is independent of Altair, as not a single line of Altair code was executed.

I set a configuration in the policy.tcl file, but it is not taking effect!

Most likely, the file has not been read yet. Try a:

```
% nc cmd vovproject sanity
```

I have a lot of log files, how can I remove the older files?

An easy way to remove files that are over 60 days old is using the vovcleanup command:

```
% nc cmd vovcleanup -proj
```

 **Note:** When preemption is heavily used, log files tend to build up.

How do I test a policy change before releasing it to production?

To test policy changes you can use the soft release mechanism. Here's a summary:

- Create a test queue.
- Set up the test to use files from the repository of the master queue.
- Test a hot file in a sandbox, identify and fix the errors before releasing it to the production domain.

I am upgrading the software - how do I suspend Accelerator from dispatching jobs?

Typically, to minimize the impact of upgrading the overhauling the system, the vovserver is stopped from dispatching new jobs, while jobs that are running are allowed to complete on the vovtasker. There is more than one way to do this: Cold Upgrade, Hot Upgrade and Rolling Hot Upgrade. For more information and instructions, refer to [Upgrade Accelerator](#).

HPC Advice

This section provides recommendations to obtain the maximum performance from your Accelerator. As Accelerator is a fast system, fine-tuning performance may only be needed when running several hundreds of thousands of jobs daily.

Use the Latest Altair Accelerator Release

The performance of the Accelerator scheduler is frequently updated. Using the most current version is recommended.

Use the vwn Wrapper

The wrapper vwn (alias for vw -d) is a faster wrapped because it avoids communication with vovserver. The regular vw checks the timestamp of the outputs after the job is done, whereas vwn does not. An example is shown below:

```
% nc run -wrapper vwn -array 100 sleep 0
```

To further push performance of the scheduler, you may want to use two options:

- -nolog: this disables the creation of the log file
- -nodb: this disables the logging of the job execution used for adding job info to the database

```
% nc run -wrapper vwn -nodb -nolog -array 100 sleep 0
```

- The benefit of using vwn is speed.
- The disadvantage is that jobs that require the -wl option cannot be run. However, this disadvantage may be not be significant, as -wl adds a relatively high load for what it does: -wl requires an extra *notify client* to handle the event generated when the job terminates.

Reduce the FairShare Window

When running millions of jobs per day, it is not important to keep a long FairShare history. Typically, a window of 2 to 5 minutes tracks sufficient history. An example follows:

```
% nc cmd vovfsgroup modrec /some/fs/tree window 2m
```

Reduce the autoForget Times

By forgetting jobs more quickly, the memory image of vovserver is kept smaller. An example is shown below:

```
# In policy.tcl
set config(autoForgetValid) 3m
set config(autoForgetFailed) 1h
set config(autoForgetOthers) 1h
```

Disable Wait Reasons

If analyzing what causes *wait time* in the workload, the wait reason analysis can be disabled as shown below:

```
# In policy.tcl
set config(enableWaitReasons) 0
```

Wait time analysis can then be re-enable as needed as shown below:

```
% nc cmd vovsh -x 'vtk_server_config enableWaitReasons 1'
### collect some data for a few minutes, then
% nc cmd vovsh -x 'vtk_server_config enableWaitReasons 0'
```


Disable File Access

Disabling file access is mostly a high-reliability option. By disabling file access, the vovserver never looks at any of the files in the user workspaces, which avoids the risk of disk slowness or disk unavailability. An example is shown below:

```
% nc cmd vovsh -x 'vtk_server_config disablefileaccess 2'
```

Reduce Update Rate of Notify Clients

Notify clients, clients that are tapping the event stream from vovserver (such as `nc gui`, `voveventmon` or `nc run -wl`), are updated immediately in the inner loop of the scheduler. If the environment includes hundreds of such clients, it may be beneficial to slow down the update rate by setting the parameter `notifySkip`. The default value is 0: no skip. Typically, the more events that take place, the more events that can be skipped without notice. For example, if several events are taking place, setting `notifySkip` to 100, fewer updates may not be noticed. If the number of events is small, a one-second delay may be noticed in some updates of the GUI. skipped without notice.

 **Note:** Regardless of the setting, the maximum time between updates is one second.

```
# In policy.tcl
set config(notifySkip) 100
```

NVIDIA™ GPUs Support in Accelerator

If you have machines with multiple GPUs, you can harness the power of those devices by following these guidelines, which have been tested with up to 8 GPUs per machine.

1. Start a "vovtaskerroot" in each machine with one consumable hardware resource for each of the GPUs on that machine. Call these resources GPU:Tesla<N> where N is an index starting from 0.

```
# In taskers.tcl
set res4gpus "GPU:Tesla0/1 GPU:Tesla1/1 GPU:Tesla2/1 GPU:Tesla3/1"
vtk_tasker_define nv001 -resources $res4gpus
vtk_tasker_define nv002 -resources $res4gpus
# ...
vtk_tasker_define nvXXX -resources $res4gpus
```

2. Define job resources of type G:Tesla<J> where J is the number of GPUs that are requested by the job. For each J you need to define the maps from the job resource to the HW resources of type GPU:TeslaN. For example, G:Tesla1 is easy and it needs to map to the OR of any of the available GPUs, while G:Tesla4 is also easy because it needs to map to the AND of all 4 devices.

```
# In resources.tcl
set mapOR "GPU:Tesla0/1 OR GPU:Tesla1/1 OR GPU:Tesla2/1 OR GPU:Tesla3/1"
set mapAND "GPU:Tesla0/1 GPU:Tesla1/1 GPU:Tesla2/1 GPU:Tesla3/1"
vtk_resource_map_set G:Tesla1 -max unlimited -map $mapOR
# ...
vtk_resource_map_set G:Tesla4 -max unlimited -map $mapAND
```

The other maps for G:Tesla2 and G:Tesla3 are more complex, and for larger values of N it is not feasible to use all possible combinations of devices. To help compute those maps, and to reduce the number of combinations to a workable subset, we provide a procedure called findCombinations in \$VOVDIR/scripts/hero/nvidia/hero_nvidia_resources.tcl. Feel free to copy that file into your resources.tcl file.

```
proc findCombinations { list n } {
    #
    # Recursive procedure to find all combinations of 'n' elements from 'list'.
    #
    set result {}
    set l [llength $list]
    if { $l >= $n } {
        set inc 1
        for { set i 0 } { $i < $l } { incr i $inc } {
            set elem [lindex $list $i]
            set subList [lreplace $list 0 $i]
            set subCombos {}
            if { $n > 1 } {
                set subCombos [findCombinations $subList [expr $n-1]]
                if { $l > 2 && [llength $subCombos] > 1 } {
                    ### When the combinations are too-many, use only the first
                    combo.
                    set subCombos [lrange $subCombos 0 0]
                }
                foreach subCombo $subCombos {
                    lappend result [concat $elem $subCombo]
                }
            } else {
                lappend result $elem
            }
        }
    }
}
```



```
    }
  }
  return $result
}

## Maximum number of GPUS in any of the farm machines.
## Call the GPUS "GPU:Tesla0 ... GPU:Tesla3 ..."
set MAX 8
set GPUS {}
for { set i 0 } { $i < $MAX } { incr i } {
  lappend GPUS "GPU:Tesla$i/1"
}

# A resource to count how many GPUs are in use.
vtk_resource_map_set G:TeslaNum -total unlimited

for { set i 1 } { $i <= $MAX } { incr i } {
  set options [findCombinations $GPUS $i ]
  set optionsWithParentheses ""
  set sep ""
  foreach opt $options {
    if { [llength $opt] > 1 } {
      append optionsWithParentheses "$sep ( $opt )"
    } else {
      append optionsWithParentheses "$sep $opt"
    }
  }
  set sep " OR"
  vtk_resource_map_set G:Tesla$i -total unlimited -map "G:TeslaNum#$i
( $optionsWithParentheses )"
}
```

3. Submit your workload using the wrapper `vovgpu` which is a script that interprets the "SOLUTION" computed by the scheduler and passes the selected list of devices to the application via the environment variable `VOV_GPUSET` or with the macro `@GPUSET@`. Note that `VOV_GPUSET` gives you a space-separated list of devices, while `@GPUSET@` gives you a comma-separated list.

With this setup, you can submit arbitrary workloads which request any number of GPUs.

```
% nc run -r G:Tesla1 -- vovgpu some_job_requiring_gpu -devices=@GPUSET@ -arg1 -arg2
% nc run -r G:Tesla2 -- vovgpu some_job_requiring_gpu -devices=@GPUSET@ -arg1 -arg2
% nc run -r G:Tesla3 -- vovgpu some_job_requiring_gpu -devices=@GPUSET@ -arg1 -arg2
% nc run -r G:Tesla4 -- vovgpu some_job_requiring_gpu -devices=@GPUSET@ -arg1 -arg2
```

If you are new to Accelerator, it is worth remembering that you can get project tracking if you use the `-jobproj` option. For FairShare, use the `-g` and `-sg` options in `nc run`, as in this example:

```
% nc run -r G:Tesla4 -jobproj MachineLearnAboutCats -g /bu/ai/rd vovgpu
my_ml_app
```

Simulation Scripts

This section describes job simulation scripts that emulate jobs. Such scripts are often used by developers as well as business systems analysts.

Typically, these scripts perform no real functions and do not access licenses; they emulate the appearance of actual usage. These scripts are often used to debug a system or configuration issue, test the capacity of the system, checking if the resources are available for upcoming jobs, and setting benchmarks for dispatching jobs, such as 1000 calls of `nc run /bin/date`.

Why Developers Use Simulation Job Scripts

Sometimes developers need to test a flow under realistic conditions to ensure that all settings are correct that users have access to resources, permissions and quotas, to run the jobs that they intend to run.

- Developers may not have access to the tools; they need to create simulated jobs for realistic testing in an artificial environment.
- Developers may have access, but in an earlier stage of development, it may be preferable to create placeholder tools, thus avoiding the use and cost of licenses.

Using Job Script Simulations for Troubleshooting and Planning

Running tests with simulated jobs can help identify hardware bottlenecks or other system limitations. Using test scripts with proportional values help generate profiles very quickly, such as usage over time. Such scripts can be used with scaled memory/time requirements, such as 1 Megabyte of memory of a test script represents 1 Gigabyte represents 1 Megabyte of actual usage, or 1 minute of a test script represents 1 hour of actual usage.

For more basic flows in which each stage consists of similar types of jobs, test scripts may not be needed. However, for more complex flows with jobs that have different characteristics and dependencies, estimating the longest path, how often job requirements result in conflict and so on, are difficult to estimate without running tests that provide results to analyze.

```
% sleep x  
% cp aa bb  
% vovmentime
```

Guidelines for Simulation Job Scripts

Frequently Used Code


- `array`
- `cp file1file2`: Emulates I/O data transfer.



Note: To successfully view a data transfer on a job profile, very large files must be used; transactions and other usages must continue at least one minute to be visible.

- `sleep x`: Do nothing during the specified time `x`. For tests and evaluations, it is best to include a random number generator. Used alone, sleep jobs complete at known, precise times - based on the specified timing, several jobs could complete simultaneously, which does not occur in actual job runs. For information about job profiles, refer to *Job Profiling* in the Altair Accelerator User Guide..
- `vovmetime`: Allocates memory, also uses CPU.
- `vtool`: Used for calling licenses. `vtool` can be used to emulate calling licenses. For information, refer to *Wrap Unlicensed Tools* in the Altair Monitor Administrator guide.

Simple examples of scripts

 **Note:** Using the sleep command alone may cause unrealistic behaviors, such as all jobs completing at the same time. Due to the scheduling of jobs and the availability of resources exact timing is unlikely. For more realistic behaviour, including a random variation of timing is recommended.

bash version

```
% #!/bin/bash
% dur=$(( $RANDOM % 5 ) + 1 )s
% echo "Sleeping for $dur"
% sleep $dur
```

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

Disable Regular User Login

This section provides guidelines to disable the ability for those with the USER level of privilege to log onto selected tasker machines. Most often, for better throughput, this is applied to use selected machines as part of a computing resource pool exclusively through Accelerator.

 **Note:** Disabling the login to a vovtasker is not a normal or supported use of VOV functionality.

Disabling user login to the selected tasker machines is done in two phases:

1. Disable user logins
2. Set up vovtsd

Phase 1

1. Disable all user logins except for the superuser or root on the selected machine.
2. Create the file `/etc/nologin`. The content of the file will be the message the users receive when they try to login.

An example of `/etc/nologin` on host `h01` is shown below:

```
% cat /etc/nologin
Login disabled. Contact admin for help.
```

When a regular (non-root) user tries to login, the message shown to the user will be as follows:

```
% rlogin -l john h01
john's Password:
Login disabled. Contact admin for help.

login:
```

Phase 2: Set Up `vovtsd` When login is disabled for the USER level and VOV ADMIN is typically not a root user, it is not possible to use `rsh` or `ssh` to either start or stop taskers from remote machines, such using the command `vovtaskermgr`. In this scenario, `vovtsd` can be used to manage the tasker machines remotely.

3. Log onto the machine as `root`, switch to VOV ADMIN and then start `vovtsd`.
4. From a remote machine, start or stop a tasker on this machine using a previously used method, such as using the command `vovtaskermgr` or `ncmgr reset`, the GUI or a browser.


```
% su - vncadmin
% vovtsd -normal
```

5. Step 4 assumes that the shell for user "vncadmin" is set up to run the VOV software. Some accounts do not have this setup. In that case, the `vovboot` script could be used to start `vovtsd`.

```
% su - root
% /full/path/to/vov/installation/common/scripts/vovboot vovtsd -normal
```

To start `vovtsd` at boot time, consider deployment of the `S99vovtsd` script, a copy of which can be found in the directory `$VOVDIR/etc/boot/S99vovtsd`. Copy this script into `/etc/rc3.d/S99vovtsd` and customize it to fit the installation requirements.

- Ensure that `vovtsd` is running.
- Automatically restart `vovtsd` on reboot of the machine. This enables the machine to provide continuous computing power without having to log in as root and manually start `vovtsd`.

 **Note:** If `vovtsd` is already running on a host and starting another host is tried, that second host will not start because the port is already occupied; starting `vovtsd` on a regular basis is a good way to ensure it is always running. Keeping the host running can be done with a cron job as shown in the example below:

```
% su - vncadmin% crontab -e
# Start vovtsd every thirty minutes
1,31 * * * * /full/path/to/VOV/common/scripts/vovboot vovtsd -normal > /
dev/null 2>&1
```

Auxiliary Group Membership

Theory

If `VOV_USE_INITGROUPS` is set, the subtasker calls `initgroups()`. This is an OS call that sets all (or max 16) auxiliary groups. The resulting list of groups is not cached. Another job will call `initgroups()` again.

The default is to not call `initgroups` because it may load the name services too much.

By default, the vovtasker calls the external utility `vovgetgroups`, which uses the value of `VOV_ALARM` to decide how long to wait for a reply (default 10 seconds). The `VOV_USE_VOVGETGROUPS` environment variable can be used to control this behavior:

Set to 0 to disable the call to the external utility and use the `getgrent()` POSIX API function to find all groups that are valid for a user. If there are more than 16, the list is truncated to the first 16. The list is cached by vovtasker, so only the first job for a user causes traffic with the name services. This is only recommended in small environments, as this method can create significant delays, and even blocking conditions, in complex environments (e.g. Linux with LDAP).

Set to 2 to continue to use the external utility, but instruct the utility to call the `getgrent()` POSIX API function instead of the default call to `getgrouplist()`. This is mainly for debugging purposes, since this mode of operation results in slower processing of group information.

History

Prior to 2016.09 & 2015.09u8

If `VOV_USE_VOVGETGROUPS` was set to any value, when a tasker needs to get group data it will use the `vovgetgroups` external utility (a separate executable). This utility is robust to LDAP errors or timeouts which would otherwise cause the `getgrent` library call to hang indefinitely (and block the tasker from issuing further jobs). Prior to customers switching to Centos6.x and SSSD name service, the use of `VOV_USE_VOVGETGROUPS` was recommended. After the switch to Centos6.x/SSSD, a bug was found that prevented all groups from being fetched. Switching to `VOV_USE_INITGROUPS=1` and leaving `VOV_USE_VOVGETGROUPS` unset appeared to fix the problem, but at the probable cost of reduced performance and increased name service load.

2016.09 & 2015.09u8 and Later Versions

If `VOV_USE_VOVGETGROUPS` was set to any value other than 1, it would behave like pre 2016.09 code and use `getgrent()`. If `VOV_USE_VOVGETGROUPS` was set to 1, it would use `getgrouplist()`, which is a newer utility (but still old) to get group information with higher performance.

The downside to setting `VOV_USE_VOVGETGROUPS=1` in 2016.09 is that there may be some off-beat OS's that don't support it. However, it seems to be faster, work with SSSD, and doesn't load the name service as much.

The recommendation based on the review of the history and the code is the following:

- Use `VOV_USE_VOVGETROUPS=1` and leave `VOV_USE_INITGROUPS` unset if you are on <2015.09u8 earlier and not using CentOS6.6 with SSSD (uses non blocking `getgrent`)
- Leave `VOV_USE_VOVGETGROUPS` unset and set `VOV_USE_INITGROUPS=1` if you are on < 2015.09u8 and want to use CentOS6.6/SSSD (uses an extra group `init` & `getgrent`)
- Set `VOV_USE_VOVGETGROUPS=1` and leave `VOV_USE_INITGROUPS` unset if you are on 2016.09 or >2015.09u7 and running a common OS (non blocking, `getgrouplist`)
- Set `VOV_USE_VOVGETGROUPS=1` and leave `VOV_USE_INITGROUPS` unset if you are on 2016.09 or >2015.09u7 and running an uncommon OS (non blocking, `getgrent`).

If both are set then VOV_USE_VOVGETGROUPS dominates.

Troubleshooting

Answers to common questions when using Monitor.

The Server Doesn't Start

1. Make sure you have a valid RLM license. Type:

```
% rlmstat -a
```

2. Check if the server for your project is already running on the same machine. Do not start an Accelerator project server more than once.

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

3. Check if the server is trying to use a port number that is already used by another vovserver or even by 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.
4. Check if the server is trying to use an inactive port number that cannot be bound. This can happen when an application, perhaps the server itself, terminates without closing all its sockets.

The server will exit with a message similar to the following:

```
...more output from vovserver...
vs52 Nov 02 17:34:55          0          3          /home/john/vov
vs52 Nov 02 17:34:55 Adding licadm@venus to notification manager
vs52 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 VOV server 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?
```

- a) If this happens, 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 you can kill it, for example with `vovkill`.

```
% vovkill pid
```

- c) Restart the server.

5. You run the server as the Accelerator administrator user. Please check the ownership of the file `security.tcl` in the server configuration directory `vnc.swd`.

UNIX Taskers Don't Start

Accelerator normally relies on remote shell execution to start the taskers, using either `rsh` or `ssh`.

- If using `rsh` try the following:

```
% rsh host vovarch
```

where `host` is the name of a machine on which there are problems starting a tasker.

This command should return a platform dependent string and nothing else. Otherwise, there are problems with either with the remote execution permission or the shell start-up script.

- If the error message is similar to "Permission denied", check the file `.rhosts` in your home directory. The file should contain a list of host names from which remote execution is allowed. See the manual pages for `rsh` and `rhosts` for details. You may have to work with your system administrators to find out if your network configuration allows remote execution.
- If using `ssh`, perform the test above but use `ssh` instead of `rsh`. For more details about `ssh` see SSH Setup in the *VOV Subsystem Administrator Guide*.
- If you get extraneous output from the above command, the problem is probably in your shell start-up script. If you are a C-shell user, check your `~/ .cshrc` file. The following are guidelines for a remote-execution-friendly `.cshrc` file:

- # Echo messages only if the calling shell is interactive. You can test if a shell is interactive by checking the existence of the variable `prompt`, which is defined for interactive shells. Example:

```
# Fragment of .cshrc file.
if ( $?prompt ) then
echo "I am interactive"
endif
```

- # Many `.cshrc` scripts exit early if they detect a non interactive shell. It is possible that the scripts exit before sourcing `~/ .vovrc`, which causes Accelerator to not be available in non-interactive shells. Compare the following fragments of `.cshrc` files and make sure the code in your file works properly:

The following example will not work properly for non-interactive shells:

```
if ( $?prompt ) exit
source ~/.vovrc
```

This example is correct, source `.vovrc` and then check the prompt variable:

```
source ~/.vovrc
if ( $?prompt ) exit
```

This example is also correct:

```
if ( $?prompt ) then
# Define shell aliases
...
endif
source ~/.vovrc
```

Do not apply `exec` to a sub shell. This will cause the `rsh` command to hang.

```
# Do not do this in a .cshrc file
exec tcsh
```

License Violation

Accelerator is licensed by restricting the number of taskers. This is the number of all unique hosts that run taskers in all instances of Accelerator servers that use the same license.

You can find out the capacity of your license with the following command:

```
% rlmstat -avail
```

The file `$VOVDIR/../../../../vnc/vnc.swd/taskers.tcl` defines the list of hosts that are managed by the server. Make sure the number of tasker hosts is within the license capability.

Crash Recovery

In the event of a crash or failover, you can find a checklist of what to do on the System Recovery page.

This address can found using the command:

```
nc cmd vovbrowser -url /cgi/sysrecovery.cgi
```

Deprecated Commands

This section lists the recently deprecated and obsoleted Accelerator commands and options, and the corresponding current commands and options.

Deprecated Accelerator commands and options


Deprecated/Obsoleted Command or Option	Current Command
lmresources	Current command: lmhandlesall. For more information, refer to Match Jobs to Handles
-J	Current command: -N. These options are used to set JobName. For more information, refer to Jobname Attribute . Also, see Node Fields for all the fields.
ccsub	Current command: The option -clearcase in nc run. For more information, refer to Job Resources

LSF Emulation

This document is intended for those who have been using the Platform LSF batch system and are now moving over to the Accelerator system.

The following information describes the available resources, specific scripts, and guidelines of using those resources.

The Altair Accelerator installation includes scripts that are designed to assist the transition from the Platform LSF batch system to the Accelerator system. The LSF scripts are provided in addition to the scripts that support the ability to customize Accelerator capabilities as well as add new commands.

 **Note:** The options supported by emulation these scripts are not 100% complete; they do support many of the applications that are frequently used.

The LSF emulation commands are also useful with EDA programs such as Cadence ADE and Altos Liberate that do not support Accelerator directly.

The emulated commands include the following:

- bhist
- bhosts
- bjobs
- bkill
- bmgrouop
- bpeek
- bqueues
- bstat
- bsub
- lshosts
- lsid

The scripts that emulate the above commands are available in the directory `$VOVDIR/scripts/lsfemulation`, which is not in the PATH in the default setup. This setup avoids collision with the platform scripts and commands.

The emulated commands are added to the path by adding the Altair Accelerator named environment LSFEMUL. The LSFEMUL environment setup is installed in `$VOVDIR/etc/environments/LSFEMUL.*`.

```
% ves +LSFEMUL
% bsub sleep 10
% bjobs -a
```


For maximize the benefits of using Accelerator it is strongly encouraged to use the native Accelerator commands for new projects and to migrate existing projects from LSF Platform commands and Altair Accelerator scripts to the Accelerator commands.

Configure Resource Mapping

The resources used by LSF, expressed by the -R option in bsub, are significantly different from the resource maps in Accelerator. LSF bsub emulation allows multiple -R directives.

To map resources from one system to the other, customize the file `$VOVDIR/local/lsfemulation/config.bsub.tcl`

This file contains a set of assignments to the Tcl arrays `MAP_LSF2NC()` and `MAP_RUSAGE()`.

 **Note:** This file is used only by the Altair Accelerator bsub emulator.

An example for `config.bsub.tcl` can be found in the usual location for configuration files, a subdirectory of `$VOVDIR/etc/config`:

File: `$VOVDIR/etc/config/lsfemulation/config.bsub.tcl`

```
#
# Sample configuration of the bsub emulation.
#
# This file must be placed in $VOVDIR/local/lsfemulation/config.bsub.tcl
#

# If you want support for exclusive access to machines (option -x)
# you need to:
# 1. Uncomment the line below 'set bsubopt(percent) 1'
# 2. Make sure all taskers offer the resource PERCENT/100
# 3. Make sure all jobs request at least PERCENT/1 (see vnc_policy.tcl)
set bsubctrl(percent) 1

# Set this to 1 to cause bsub to always send email
set bsubctrl(alwaysmail) 1

# Truncate emailed log files after this many bytes
# negative values (e.g. -1) mean mail the whole file (BEWARE)
# zero means accept the default (65536 bytes)
set bsubctrl(logmax) 0

# Emulation transforms -m hostnames into an OR expression,
# which can slow down the NC scheduler if too complex.
# Hosts after this count are silently dropped to avoid slow scheduling
set bsubctrl(moptmax) 6

# Map LSF 'select' resources into NC resources.
# select[rhel4] -> "linux"
set MAP_LSF2NC(rhel3) "linux"
set MAP_LSF2NC(rhel4) "linux"
set MAP_LSF2NC(rhel5) "linux"
set MAP_LSF2NC(RH4_64) "linux x86_64"

# Map LSF 'rusage' resources into NC resources,
# typically resources of type License:
# Example:
# rusage[dc=1] -> "License:Design-Compiler"

set MAP_RUSAGE(dc) "License:Design-Compiler"
set MAP_RUSAGE(pt) "License:PrimeTime"
set MAP_RUSAGE(drc) "License:lic_drc"
set MAP_RUSAGE(lvs) "License:lic_lvs"
set MAP_RUSAGE(erc) "License:lic_erc"
```

Emulate the LSF Report in the Output Log

Some legacy scripts expect some LSF specific lines in the log file of a job. This can be achieved with a post-command that adds those lines to the log. An example of such command is `post_job_report.sh`.

```
#!/bin/csh -f
# -*- Tcl -*- \
    exec vovsh -f $0 $*:q

set usage "
Description:
    post_job_report.sh

    Used for some jobs submitted with the bsub emulator:

Example:
% ves +LSFEMUL
% bsub -Ep $VOVDIR/etc/post/post_job_report.sh -J test.lsf_with_jobreport cal 2015
"

if { $argv == {} } {
    VovPrintUsage $usage
}

source $env(VOVDIR)/tcl/vtcl/vovlsfemulib.tcl

set jobId      [lindex $argv 0]
set logFileName [lsfEmuGetJobLogFileName $jobId]
set report      [lsfEmuFmtJobReport $jobId]

if { $logFileName ne "" } {
    VovMessage "Adding job report to $logFileName"
    set fp [open $logFileName "a"]
    puts $fp [lsfEmuFmtJobReport $jobId]
    close $fp
} else {
    VovMessage "No log file found for job $jobId\n$report"

    set whyOld ""
    set whyNew "Cannot find a log file for this job $jobId"
    if { [catch {set whyOld [vtk_prop_get $jobId WHY]}] } {
        set why $whyNew
    } else {
        set why "$whyOld\n$whyNew\n$report"
    }
    catch {vtk_prop_set $jobId WHY $why}
}

exit 0
```

The post command can be specified with the option `-Ep` of the `bsub` emulator. For example:

```
% bsub -Ep $VOVDIR/etc/post/post_job_report.sh [OTHER OPTIONS]
... command
```

Debug the LSF Emulation Layer Usage

To debug as well as test and verify an LSF emulation script, it can be helpful to view the issued commands and the used options and values.

If the environment is set with variable `VOV_LOG_LSFEMUL` to the name of a file, all emulation commands will be logged in that file. For example:

```
% setenv VOV_LOG_LSFEMUL ~/lsfemul.log
% bsub sleep 11
% lsid
% cat $VOV_LOG_LSFEMUL
```

Legal Notices

Intellectual Property Rights Notice

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

Copyright © 1986-2023 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of 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.

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 Simulation Products

Altair® AcuSolve® ©1997-2023

Altair Activate® ©1989-2023

Altair® Battery Designer™ ©2019-2023

Altair Compose® ©2007-2023

Altair® ConnectMe™ ©2014-2023

Altair® EDEM™ © 2005-2023

Altair® ElectroFlo™ ©1992-2023

Altair Embed® ©1989-2023

Altair Embed® SE ©1989-2023

Altair Embed®/Digital Power Designer ©2012-2023

Altair Embed® Viewer ©1996-2023

Altair® ESAComp® ©1992-2023

Altair® Feko® ©1999-2023

Altair® Flow Simulator™ ©2016-2023

Altair® Flux® ©1983-2023

Altair® FluxMotor® ©2017-2023

Altair® HyperCrash® ©2001-2023

Altair® HyperGraph® ©1995-2023

Altair® HyperLife® ©1990-2023

Altair® HyperMesh® ©1990-2023
Altair® HyperSpice™ ©2017-2023
Altair® HyperStudy® ©1999-2023
Altair® HyperView® ©1999-2023
Altair® HyperViewPlayer® © 2022-2023
Altair® HyperWorks® ©1990-2023
Altair® HyperXtrude® ©1999-2023
Altair® Inspire™ ©2009-2023
Altair® Inspire™ Cast ©2011-2023
Altair® Inspire™ Extrude Metal ©1996-2023
Altair® Inspire™ Extrude Polymer ©1996-2023
Altair® Inspire™ Form ©1998-2023
Altair® Inspire™ Mold ©2009-2023
Altair® Inspire™ PolyFoam ©2009-2023
Altair® Inspire™ Print3D ©2021-2023
Altair® Inspire™ Render©1993-2023
Altair® Inspire™ Studio ©1993-2023
Altair® Material Data Center™ ©2019-2023
Altair® MotionSolve® ©2002-2023
Altair® MotionView® ©1993-2023
Altair® Multiscale Designer® ©2011-2023
Altair® nanoFluidX® ©2013-2023
Altair® OptiStruct® ©1996-2023
Altair® PolEx™ ©2003-2023
Altair® PSIM™ © 2022-2023
Altair® Pulse™ ©2020-2023
Altair® Radioss® ©1986-2023
Altair® romAI™ © 2022-2023
Altair® S-FRAME® © 1995-2023
Altair® S-STEEL™ © 1995-2023
Altair® S-PAD™ © 1995-2023
Altair® S-CONCRETE™ © 1995-2023
Altair® S-LINE™ © 1995-2023

Altair® S-TIMBER™ © 1995-2023

Altair® S-FOUNDATION™ © 1995-2023

Altair® S-CALC™ © 1995-2023

Altair® S-VIEW™ © 1995-2023

Altair® Structural Office™ © 2022-2023

Altair® SEAM® © 1985-2023

Altair® SimLab® ©2004-2023

Altair® SimLab® ST © 2019-2023

Altair SimSolid® ©2015-2023

Altair® ultraFluidX® ©2010-2023

Altair® Virtual Wind Tunnel™ ©2012-2023

Altair® WinProp™ ©2000-2023

Altair® WRAP™ ©1998-2023

Altair® GateVision PRO™ ©2002-2023

Altair® RTLvision PRO™ ©2002-2023

Altair® SpiceVision PRO™ ©2002-2023

Altair® StarVision PRO™ ©2002-2023

Altair® EEvision™ ©2018-2023

Altair Packaged Solution Offerings (PSOs)

Altair® Automated Reporting Director™ ©2008-2022

Altair® e-Motor Director™ ©2019-2023

Altair® Geomechanics Director™ ©2011-2022

Altair® Impact Simulation Director™ ©2010-2022

Altair® Model Mesher Director™ ©2010-2023

Altair® NVH Director™ ©2010-2023

Altair® NVH Full Vehicle™ © 2022-2023

Altair® NVH Standard™ © 2022-2023

Altair® Squeak and Rattle Director™ ©2012-2023

Altair® Virtual Gauge Director™ ©2012-2023

Altair® Weld Certification Director™ ©2014-2023

Altair® Multi-Disciplinary Optimization Director™ ©2012-2023

Altair HPC & Cloud Products

Altair® PBS Professional® ©1994-2023

Altair® PBS Works™ © 2022-2023

Altair® Control™ ©2008-2023

Altair® Access™ ©2008-2023

Altair® Accelerator™ ©1995-2023

Altair® Accelerator™ Plus ©1995-2023

Altair® FlowTracer™ ©1995-2023

Altair® Allocator™ ©1995-2023

Altair® Monitor™ ©1995-2023

Altair® Hero™ ©1995-2023

Altair® Software Asset Optimization (SAO)™ ©2007-2023

Altair Mistral™ ©2022-2023

Altair® Grid Engine® ©2001, 2011-2023

Altair® DesignAI™ ©2022-2023

Altair Breeze™ ©2022-2023

Altair® NavOps® © 2022-2023

Altair® Unlimited™ © 2022-2023

Altair Data Analytics Products

Altair Analytics Workbench™ © 2002-2023

Altair® Knowledge Studio® © 1994-2023

Altair® Knowledge Studio® for Apache Spark © 1994-2023

Altair® Knowledge Seeker™ © 1994-2023

Altair® Knowledge Hub™ © 2017-2023

Altair® Monarch® © 1996-2023

Altair® Panopticon™ © 2004-2023

Altair® SmartWorks™ © 2021-2023

Altair SLC™ ©2002-2023

Altair SmartWorks Hub™ ©2002-2023

Altair® RapidMiner® © 2001-2023

Altair One™ ©1994-2023

Third Party Software Licenses

AcuConsole contains material licensed from Intelligent Light (www.ilight.com) and used by permission.

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

Technical Support

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

Altair One Customer Portal

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

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

Altair Training Classes

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

Telephone and E-mail

If you are unable to contact Altair support via the customer portal, you may reach out to the technical support desk via phone or e-mail. You can use the following table as a reference to locate the support office for your region.

When contacting Altair support, please specify the product and version number you are using along with a detailed description of the problem. It is beneficial for the support engineer to know what type of workstation, operating system, RAM, and graphics board you have, so please include that in your communication.

Location	Telephone	E-mail
Australia	+61 3 9866 5557 +61 4 1486 0829	anz-pbssupport@altair.com
China	+86 21 6117 1666	pbs@altair.com.cn
France	+33 (0)1 4133 0992	pbssupport@europe.altair.com
Germany	+49 (0)7031 6208 22	pbssupport@europe.altair.com
India	+91 80 66 29 4500 +1 800 208 9234 (Toll Free)	pbs-support@india.altair.com
Italy	+39 800 905595	pbssupport@europe.altair.com
Japan	+81 3 6225 5821	pbs@altairjp.co.jp
Korea	+82 70 4050 9200	support@altair.co.kr

Location	Telephone	E-mail
Malaysia	+91 80 66 29 4500 +1 800 208 9234 (Toll Free)	pbs-support@india.altair.com
North America	+1 248 614 2425	pbssupport@altair.com
Russia	+49 7031 6208 22	pbssupport@europe.altair.com
Scandinavia	+46 (0) 46 460 2828	pbssupport@europe.altair.com
Singapore	+91 80 66 29 4500 +1 800 208 9234 (Toll Free)	pbs-support@india.altair.com
South Africa	+27 21 831 1500	pbssupport@europe.altair.com
South America	+55 11 3884 0414	br_support@altair.com
United Kingdom	+44 (0)1926 468 600	pbssupport@europe.altair.com

See www.altair.com for complete information on Altair, our team and our products.

Index

Special Characters

\$subdirectories, cleaning [66](#)

A

Accelerator and ClearCase: nc run -clearcase [101](#)

Accelerator quick start [9](#)

Accelerator User Guide [4](#)

access to help [7](#)

accessing resources for jobs [101](#)

active component, distributed parallel job [49](#)

ADE, Cadence [147](#)

advanced information [111](#)

allocate resources, FairShare [63](#)

autoforget jobs [36](#)

autoforget log files [36](#)

autoforget parameters [36](#)

automatic rerunning of failed jobs [38](#)

auxiliary group membership [142](#)

B

backwards compatibility [146](#)

bhist, emulated [147](#)

bhosts, emulated [147](#)

bjobs, emulated [147](#)

bkill, emulated [147](#)

browser-based setup [27](#)

browser, submitting jobs from [27](#)

bsub, emulated [147](#)

C

c-shell, TCSH user setup [16](#)

caching, nc list [75](#)

CGROUPS for jobs [105](#)

change the default queue with nc_queue [45](#)

choosing the FairShare group [63](#)

class, job [41](#)

clean up log files [66](#)

ClearCase view [] [101](#)

command line interface [12](#)

command line, run jobs [18](#)

control the environment [98, 99](#)

CPU progress and run status indicators [70](#)

CPUPROGRESS, percentage of CPU time [70](#)

CPUTIME, total CPU time accumulated [70](#)
cross-platform job runs [101](#)

D

debug jobs example [68](#)
debug jobs without running Accelerator [68](#)
deep, clean option [66](#)
default output of nc run [25](#)
deprecated Accelerator commands and options [146](#)
detailed job information [28](#)
disable handle matching [110](#)
disable regular user login [140](#)
distributed parallel component rank and resources [50](#)
distributed parallel properties [50](#)
distributed parallel slot timeout [52](#)
distributed parallel support [49](#)

E

emulate job, test script [138](#)
emulation, LSF [147](#)
environment control [98, 99](#)
environment, cross-platform job [101](#)
example, submit a single job [25](#)
example, submit multiple jobs [26](#)
exclamation-point (bang) operator [12](#)

F

FairShare group, managed by policy or vovsgroup [63](#)
FairShare group, selected by user [63](#)
FairShare groups [63](#)
field reference [123](#)
find available jobclasses [41](#)
forget jobs [35](#)
format strings [123](#)
frequently asked questions and troubleshooting tips [131](#)

G

get detailed information about a job [29](#)
get job information [28](#)
graphical view of job progress and results [30](#)
groups [63](#)

H

handle matching, disable [110](#)

help, Accelerator [7](#)
help, quick start [9](#)
HPC advice [134](#)

I

icons [80](#)
incompatible fields [126](#)
integration with Monitor-basic [4](#)
interactive jobs [57](#)
interactive jobs restrictions and consequences [61](#)
invoke the GUI [79](#)

J

job arrays [40](#)
job i/o profiling [73](#)
job placement policies [64](#)
job profiling [71](#)
job resource plots [95](#)
job resources [101](#)
job running overview [18](#)
job runtime - monitoring and profiling [70](#)
job setup options [40](#)
job status [30](#)
job status notification [78](#)
job submission arguments [56](#)
job, emulate with script [] [138](#)
job, waiting for resources [46](#)
jobclasses [41](#)
jobname attribute [40](#)
jobs, rerun [37](#)
jobs, show current information [79](#)
jobs,pending [46](#)
jobs,queued [46](#)
jobs,scheduled [46](#)

L

license handle, match to jobs [108](#)
listing jobs [75](#)
log file, interactive job [57](#)
log files [42](#)
LSF
 emulation [147](#)
LSF emulation [147](#), [147](#)
LSF migration [129](#)
LSF, platform [147](#)

M

- MAILTO notifications, job status [78](#)
- MAILTO properties [78](#)
- manage jobs [32](#)
- match jobs to handles [108](#)
- metrics, scheduler [79](#)
- migrate from LSF [129](#)
- modify running jobs [57](#)
- modify scheduled jobs [59](#)
- monitor job, RAM, CPU and children [70](#)
- monitor resource availability [92](#)
- monitor resource utilization [92](#)
- monitor the workload [75](#)
- Monitor-basic integration [4](#)
- monitoring jobs, taskers and resources [88](#)
- multiphase support [54](#)
- multiple jobs, submit [26](#)

N

- nc clean [66](#)
- nc forget [35](#), [35](#)
- nc jobclass [41](#)
- nc list [75](#), [75](#)
- nc rerun [38](#)
- nc run [18](#)
- nc stop [33](#)
- nc summary [77](#)
- nc wait [32](#), [32](#)
- nc why [46](#), [46](#)
- nc_gui [79](#)
- NC_QUEUE environment variable [44](#)
- ncmgr command [10](#)
- ncmgr start [44](#)
- node fields [112](#)
- notification of job status [78](#)
- nozap, clean option [66](#)
- NUMA control and CPU affinity [104](#)
- NVIDIA GPUs support in Accelerator [136](#)

O

- online help [7](#)
- OpenMPI support [54](#)

P

- PDF, access [7](#)

- policy [63](#)
- post-condition [42](#)
- pre-command and post-command job condition [42](#)
- pre-condition [42](#)
- pre-pending and appending arguments in a job submission [56](#)
- predicates, in selection rules [126](#)
- priority [47](#)

Q

- queue name, default [9](#)
- queue selection [44](#)
- quick reference [14](#)
- quick start help [9](#)

R

- request a license before executing jobs [102](#)
- rerun jobs [37](#)
- resource availability [92](#)
- resource plots [92](#)
- resource statistics [90](#)
- resource utilization [92](#)
- resource/handle matching [110](#)
- run jobs with CLI commands [18](#)
- running interactive jobs [57](#)
- RUNSTATUS field values [70](#)

S

- sanity check for vovserver [139](#)
- schedule job submission [44](#)
- scheduled jobs [46](#)
- scheduler metrics [79](#)
- script, emulate jobs [138](#)
- select a named environment [99](#)
- select a queue name [45](#)
- select FairShare group [63](#)
- selection rules [126](#)
- set status [31](#)
- set the range for VovId [111](#)
- setting up the user shell [16](#)
- showing the hosts/taskers [82](#)
- simulation scripts [138](#), [138](#)
- single job submit example [25](#)
- size specification [123](#)
- specify FairShare subgroup, allocate computing resources [63](#)
- start a queue [44](#)

- statistics overview [90, 90](#)
- status of a set [31](#)
- status of jobs [30](#)
- stop jobs [33](#)
- stuck job, troubleshoot with LASTCUPROGRESS [70](#)
- submission of jobs with pre-condition or post-condition [42](#)
- submit job, with jobclass [41](#)
- submit jobs [18](#)
- submit jobs from the browser [27](#)
- submit jobs using jobclasses [41](#)
- submit jobs with CLI commands [25](#)
- submitted job information display [27](#)
- summary of all jobs [77](#)
- system overview [4](#)

T

- time specifications [124](#)
- troubleshoot stuck jobs [70](#)
- troubleshooting [143](#)
- troubleshooting - crash recovery [145](#)
- troubleshooting - license violation [145](#)
- troubleshooting - server doesn't start [143](#)
- troubleshooting - UNIX taskers don't start [144](#)

U

- use Accelerator help [7](#)
- use containers for jobs [107](#)
- use vovselect for querying [84](#)
- user setup: bourne shell, k-shell, z-shell, bash [16](#)
- user setup: c-shell, TCSH [16](#)
- user setup: Windows command shell [16](#)
- user shell setup [16](#)
- using different resources and jobclasses [51](#)
- using jobclasses [41](#)
- using snapshot with named environment [99](#)
- utility vovresreq: request a license before executing jobs [102](#)

V

- verify your setup [17](#)
- view job progress and results [27](#)
- vnc_logs, cleaning [66](#)
- vnc, default name of queue [9](#)
- VNC: quick reference [14](#)
- vov [16](#)
- VOV [9, 12, 14, 111, 123, 124, 147](#)

VOV_DP_COUNT [49](#)
VOV_DP_RANK [49](#)
VOV_DP_TOPJOBID [49](#)
VOV_ENV [98](#)
VOV_JOB_DESC [41](#), [71](#)
VOV_JOBINDEX [40](#)
VOV_LIMIT_cputime [33](#)
VOV_LOG_LSFEMUL [147](#)
VOV_STDOUT_SPEC [7](#)
VOV_STOP_SIGNAL_DELAY [33](#)
VOV_STOP_SIGNALS [33](#)
vovarch [16](#)
VOVARCH [98](#)
vovbrowser [7](#)
vovbuild [7](#), [80](#)
vovcalibremt [49](#)
vovconsole [79](#)
VovDate [124](#)
VOVDIR [9](#), [16](#), [44](#), [54](#), [71](#)
vovdoc [7](#), [7](#)
vovenvutils [98](#)
vovfire [12](#)
vovfsgroup [63](#)
vovid [7](#)
VovId [111](#)
vovinit [16](#)
vovlsfemulib [147](#)
vovmemtime [33](#), [138](#)
VovMessage [147](#)
vovmetime [138](#)
vovmpirun [54](#)
vovnotifyd [78](#)
vovparallel [49](#)
vovparallel close [53](#)
VovParseTimeSpec [124](#)
VovPrintUsage [147](#)
vovproject [139](#)
vovprop [78](#)
vovproperties [101](#), [102](#)
vovrc [16](#)
vovresgrab [102](#), [104](#)
vovresourced [139](#)
vovresreq [102](#), [102](#)
vovresSetFlags [110](#)
VovScanClock [124](#)
vovselect [82](#), [104](#), [104](#)
vovserver [4](#), [7](#), [9](#), [44](#), [46](#), [68](#), [68](#), [110](#), [129](#), [139](#)

VovServer [44](#)
vovset [12](#), [123](#)
vovsetupuser [16](#)
vovsh [36](#), [111](#), [147](#)
vovshow [33](#), [63](#)
vovtasker [33](#), [68](#), [68](#), [70](#)
vovtaskermgr [139](#)
vovtaskers [82](#), [104](#)
vovversion [16](#)
vsz, cleaning [66](#)
vtk_generic_get [111](#)
vtk_jobclass_set_autoforget [36](#)
vtk_jobclass_set_max_reschedule [38](#)
vtk_prop_get [49](#), [147](#)
vtk_prop_set [147](#)
vtk_resourcemap_set [47](#), [110](#)
vtk_server_config [111](#)
vtk_time_pp [124](#)

W

wait reasons [46](#)
waiting, stopping, cleaning, debugging jobs [32](#)
what happens when jobs are running [18](#)
ww ccexec, wrapper [101](#)