



ALTAIR

ONLY FORWARD

Altair Accelerator Plus 2024.1.0

User Guide

Contents

Accelerator Plus User Guide	4
Use Accelerator Plus Help.....	6
Accelerator Plus Quick Start.....	8
Command Line Interface.....	11
User Shell Setup.....	12
Verify Your Setup.....	13
Running a Job.....	14
Submitted Job Information Display.....	14
Scheduled Jobs.....	15
Submit Jobs to Accelerator Plus.....	16
Queue Selection.....	16
Submit Jobs with CLI Commands.....	16
Job Arrays.....	18
Environment Control.....	19
Pre-Command and Post-Command Job Conditions.....	21
Use Jobclasses.....	22
Choose the FairShare Group.....	24
Job Submission Arguments.....	24
Priority.....	25
Modify Running Jobs.....	27
Interactive Jobs.....	27
Modify Scheduled Jobs.....	29
Restrictions and Consequences.....	30
Monitor the Workload.....	32
Get Detailed Information about a Job.....	32
Job Status.....	33
Job Runtime - Monitor and Profile.....	34
Job Profiling.....	35
List Jobs.....	37
Job Summary.....	39
Invoke the GUI.....	40
Icons.....	42
Show the Hosts/Taskers.....	44
Monitor Jobs, Taskers and Resources.....	49
Manage Jobs.....	51
Wait for Jobs.....	51
Stop Jobs.....	52
Forget Jobs.....	54
Clean Up Log Files.....	55
Debug Jobs without Running Accelerator Plus.....	58
Debug Jobs Example.....	58

LSF Emulation.....	60
Legal Notices	63
Intellectual Property Rights Notice.....	64
Technical Support.....	68
Index	70

Accelerator Plus is a high-performance hierarchical scheduler designed for distributed High Performance Computing (HPC) environments.

This chapter covers the following:

- [Use Accelerator Plus Help](#) (p. 6)
- [Accelerator Plus Quick Start](#) (p. 8)
- [Command Line Interface](#) (p. 11)
- [User Shell Setup](#) (p. 12)
- [Running a Job](#) (p. 14)
- [Submit Jobs to Accelerator Plus](#) (p. 16)
- [Modify Running Jobs](#) (p. 27)
- [Monitor the Workload](#) (p. 32)
- [Manage Jobs](#) (p. 51)
- [Debug Jobs without Running Accelerator Plus](#) (p. 58)
- [LSF Emulation](#) (p. 60)

Accelerator Plus is based on the patented concepts described in US Patent 9,658,893 about multi-layered resource scheduling.

The current implementation is designed to be run in conjunction with a base scheduler such as Accelerator™, or Altair PBS Professional®.

With its sub-millisecond latency, Accelerator Plus improves the throughput of difficult workloads, especially those consisting of large numbers of short duration jobs perhaps with complex dependencies, while off-loading the base scheduler. Accelerator Plus allows any user or group to have their own high-performance scheduler without requiring the intervention of the IT department. Since all computing resources are negotiated by means of the base scheduler, Accelerator Plus always obeys all policies established by IT with respect to sharing such resources.

Theory of Operation

During the initial setup, the Accelerator Plus 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 computing resources, and *notify* clients that listen for events.

A fresh instance of Accelerator Plus typically has only one persistent or permanent tasker, dedicated to launching requests to get more taskers from the underlying base scheduler, depending on the workload.

Regular clients can submit the workload, which consists of one or more jobs, or query data about jobs or system status. When a job is created, it is placed in a queued state. Queued jobs are sorted into buckets. Jobs that have the same characteristics go in the same bucket.

Each job bucket is analyzed, by an external daemon called `vovwxcd`. If a bucket is waiting for hardware resources, then the external daemon issues a request to the underlying base scheduler for resources that match that job bucket. In other words, Accelerator Plus requests from the base scheduler a tasker that can run the jobs in a specific bucket. Once the base scheduler grants the request by running a proxy job, the submitted `wx-tasker` connects back to the Accelerator Plus instance advertising the available resources. Jobs from the matching bucket begin executing without any further intervention from the base scheduler. Multiple buckets and multiple jobs from each bucket can be serviced concurrently. With a large base scheduler and a significant workload, thousands of jobs can be run concurrently.

When a job completes, the `wx-tasker` notifies the `vovserver`. The resources, both tasker-based and central, are recovered, allowing subsequent jobs (queued in the buckets) to be dispatched. When completed, the job status is updated to either `VALID` or `FAILED`.

In addition to dispatching jobs and processing their status, the `vovserver` responds to queries about system and job requests, publishes events to notify clients, and continues to process incoming job requests.

Examples of Modes of Operation

Accelerator Plus can be used in many ways. Here are some typical examples.

Single User Mode, Persistent

Here a Accelerator Plus instance is started on a dedicated compute node using a role account. Another application, for example a Jenkins build server, is used to create the workload. In this scenario, Accelerator Plus is used primarily as an efficient distributed build engine, interfacing with the base scheduler. Multiple Accelerator Plus instances can be deployed concurrently to accelerate multiple flows in the form of execution "lanes." The underlying scheduler is used to balance the resource allocation across the Accelerator Plus instances.

Single User Mode, On-Demand

Similar to the first mode but this time the Accelerator Plus instance itself is also run on the underlying batch system. Upon completion of the workload, the Accelerator Plus instance is halted and all compute resources are returned to the farm. This model is useful for occasional, self-contained resource intensive workloads.

Multi User Mode, Persistent

This mode implements full hierarchical scheduling. The Accelerator Plus instance runs on a dedicated node with a publicly known host name and port number. Multiple Accelerator Plus instances can be used concurrently to provide each team with their own scheduler. While it is possible to allocate Accelerator Plus instances on a per-project basis, the preferred allocation method is on a functional or workload basis. For example, providing an Accelerator Plus instance for each of the Design Verification, Circuit Design and Physical Design teams allows similar work flows to be grouped together on a single Accelerator Plus instance. Commonality of work flow within an Accelerator Plus instance allows more optimal tuning while sharing a common base scheduler.

Use Accelerator Plus Help

Accelerator Plus documentation is available in HTML and PDF format.

Access the Help when Accelerator Plus is Running

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

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

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

Accelerator Plus has two main commands, wx and wxmgr.

Accelerator Plus has two main commands:

- wx is used to submit, query, and stop jobs. This command can also be invoked as vnc or as nc.
- wxmgr is used to start and stop a queue, that is, an Accelerator Plus instance.

WX

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

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

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 Plus.
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) 1998-2023, Altair Engineering.

wxmgr

This program manages the vovserver for Accelerator Plus.

wxmgr: Usage Message

This program manages the vovserver for WorkloadXelerator.
Copyright (c) 1998-2021, Altair Engineering.

USAGE:

```
wxmgr help|info|reset|start|stop|cm [OPTIONS]
```

ACTIONS:

```
info  [-queue|-q <name>] [-v]
reset [-soft | -hard | -h ]
start [-dir <server_working_dir>] [-force] [-queue|-q <name>]
      [-port <port> ] [-webport <port>] [-roport <port>]
      [-dbhost <host>] [-dbroot <path>] [-dbport <port>]
      [-prod nc|wx] [-basequeue <name>]
      The default <server_working_dir> is
      <...>/wx.
      This is the parent of the configuration (.swd) directory for
      the queue.
stop  [-force] [-freeze] [-queue|-q <name>] [-writeprdir <dirname>]
      -force          Do not prompt for confirmation
      -freeze         Instruct taskers to keep running and wait for a
                      new server
      -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:

```
% wxmgr
% wxmgr -h
% wxmgr start -queue wx2
% wxmgr start -port 6699 -queue wx99
% wxmgr info
% wxmgr reset -soft
% wxmgr reset -hard
% wxmgr cm help
```

EXAMPLE TO STOP AND RESTART SERVER:

```
% wxmgr stop -freeze
% wxmgr start -force
```

```
% wxmgr stop -freeze -force -writeprdir /tmp/abc123
```

Command Line Interface

All user commands have the following structure:

```
% wx subcommand [options]
```

The subcommand is one of the following:

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

For example:

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

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

```
% wx l  
% wx li  
% wx lis  
% wx list
```

User Shell Setup

To set up your user shell with Accelerator Plus 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 Plus that is installed.

```
% vovversion  
2023.1.1
```

Running a Job

This section summarizes how to submit and run a job.

Information about submitting jobs is covered in [Submit Jobs to Accelerator Plus](#).

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

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 level option, such as

```
wx run -v 1 sleep 10 002020291
```

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

Verbose Level	Effect
9	Very verbose

Scheduled Jobs

Jobs that cannot be dispatched immediately due to a delay in acquiring the resources from the underlying scheduler, such as CPUs or software licenses, are put on the job queue.

If the base scheduler is Accelerator, then the FairShare policy in that queue primarily determines what resources (taskers) are made available to Accelerator Plus and those resources are allocated to Accelerator Plus jobs of a specific FairShare group. This effectively by-passes the Accelerator Plus FairShare algorithm. The rules described below are applicable to other base schedulers:

- 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 Plus 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 `wx summary`. 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.
-

Submit Jobs to Accelerator Plus

Queue Selection

There may be multiple instances of Accelerator Plus running at a given site. You choose which one to use with the variable `NC_QUEUE`, which by the way is the same variable also used to select the instance of Accelerator.

If you want to use the default Accelerator Plus queue (i.e. "wx"), use:

```
% unsetenv NC_QUEUE
% wx cmd vsi
...output about the default wx queue ...
```

To use a different queue, for example one called "mywx", use:

```
% setenv NC_QUEUE mywx
% wx cmd vsi
...output about the mywx queue ...
```

It is also possible to override the value of `NC_QUEUE` by using the `-q` option with the `wx` command:

Examples:

```
% wx -q mywx hosts
...
% wx -q wx hosts
...
```

Alternatively, you can use the `NC_QUEUE` variable to refer to the `setup.tcl` file in the `SWD` directory of the desired Accelerator Plus instance. This method avoids the use of the `NC_CONFIG_DIR` directory, (usually `$VOVDIR/local/vncConfig`) which may not have write permissions for ordinary users, and is also best if you are running multiple versions of the Accelerator Plus code. This method is recommended for all `WX` queues.

Example:

```
% setenv NC_QUEUE /home/bob/vov/mywx.swd/setup.tcl
```


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.

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

```
% wx run sleep 30
Fairshare= /time/users
Resources= linux64 CORES/1 RAM/500
Env       = SNAPSHOT(wx_logs/snapshots/jon/linux64/env16220.env)
Command   = vw sleep 30
Logfile   = wx_logs/20180404/132352.71209
JobURL    = http://SOMEHOST:SOMEPORT/cgi/node.cgi?id=001241627
JobId     = 001241627
```

Run a Job, Environment Specified (-e)

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

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


```
% wx run -e BASE -r RAM/333 -- sleep 30
Fairshare= /time/users
Resources= RAM/333 CORES/1
Env       = BASE
Command   = vw sleep 10
Logfile   = wx_logs/20180404/132618.92050
JobURL    = http://SOMEHOST:SOMEPORT/cgi/node.cgi?id=001241648
JobId     = 001241648
```

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
```

Default Output of wx run

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

- The [resource list](#) 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 Plus to this job. Job IDs are used as handles with many of the Accelerator Plus commands.

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.

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

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 TERMCPAP 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
% wx 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:

```
% wx run -e BASE sleep 10
...output omitted...
% wx run -e BASE+SPICE sleep 10
...output omitted...
% wx 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:

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

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 `wx run` to specify the pre- and post- conditions.

```
% wx run -pre $VOVDIR/etc/pre/pre_check.sh sleep 10
% wx 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:

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

Use 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 `wx 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 Plus administrator.

Find Jobclasses

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

```
wx: Usage Message
WX JOBCLASS:
    List classes defined for job submission
USAGE:
    % wx jobclass [OPTIONS]
OPTIONS:
    -h                -- This help
    -l                -- Long format (with description)
    -ll              -- Longer format.
    -v                -- Increase verbosity.
EXAMPLES:
    % wx jobclass
    % wx jobclass -l
    % wx jobclass -ll
```

For example:

```
% wx jobclass
1 short
2 interactive
```

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 Plus provides the Jobclass page. This page shows a table of the jobclass, 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 `wx run`.

```
% wx 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 `wx run` are parsed sequentially, so it is possible to do a command line override of the parameters set in the jobclass. For example, the following commands behave differently:

```
% wx run -C verilog -e DEFAULT -- run_sim chip
% wx 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.

Choose the FairShare Group

While Accelerator Plus supports the use of FairShare groups, their utility depends upon the underlying batch system.

If the underlying batch system is Accelerator then the recommendation is that no additional FairShare configuration (with access control lists, weights and time windows) is done in the Accelerator Plus queue. Instead Accelerator Plus just passes on the requested FairShare group on to Accelerator where the existing policies and allocations are enforced. For base schedulers other than Accelerator, the Accelerator Plus FairShare model may be used to prioritize jobs of different categories within the Accelerator Plus queue.

In Accelerator Plus, FairShare groups are managed by either the information in the `vwX.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 `wx run` with the option `-g` to select a different group.

Examples:

```
% wx cmd vovshow -users
% wx run -g /time/regression sleep 10
```

FairShare Subgroup

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.

Examples are shown below:

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

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
```

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:

- FairShare weightings are not currently supported in Accelerator Plus. For the jobs of a given user, higher priority jobs are scheduled before lower priority ones.
- 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.

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

```
% wx run -p high sleep 10  
% wx run -p 12 sleep 10  
% wx 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
```

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.

Example: rerun the job 123456 with increased priority:

```
% wx rerun -p incr 000123456
```

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 Plus. 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>	<code>-Ix</code>	<code>% wx 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>	<code>-Ir</code>	<code>% wx 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	<code>-Il</code>	<code>% date wx 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`:

```
% wx 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:

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

Modify Scheduled Jobs

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

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.

Monitor the Workload

Get Detailed Information about a Job

The command `wx` displays information about a job.

Get Detailed Information About a Job

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

```
wx: Usage Message

WX GETFIELD:
  Get one or all fields of one or more WX 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:
  -h                Show this help.  You can also get the
                   usage message by specifying no option at all.
  -v                Increase verbosity.
  -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.
  -f field          Specify field when giving multiple jobIDs
  -showid           Show jobId
  -s                Same as -showid
  -sep STRING       Use STRING as separator (default is a single space)
  -tab              Use a TAB character as separator.

EXAMPLES:
  % wx getfield -h
  % wx getfield 01234455
  % wx getfield 00123445 jobclass
  % wx getfield ! status
  % wx getfield -J JOBNAME
  % wx getfield 01234455 0123458 -f jobclass
  % wx getfield -s 01234455 0123458 -f jobclass
```


Examples:

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


Job Status

In Accelerator Plus, 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.
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. <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> Note: This status occurs rarely and tends to be hard to observe.</div>

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

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

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 = $1 \ll 20$ 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 Plus 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 Plus 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:

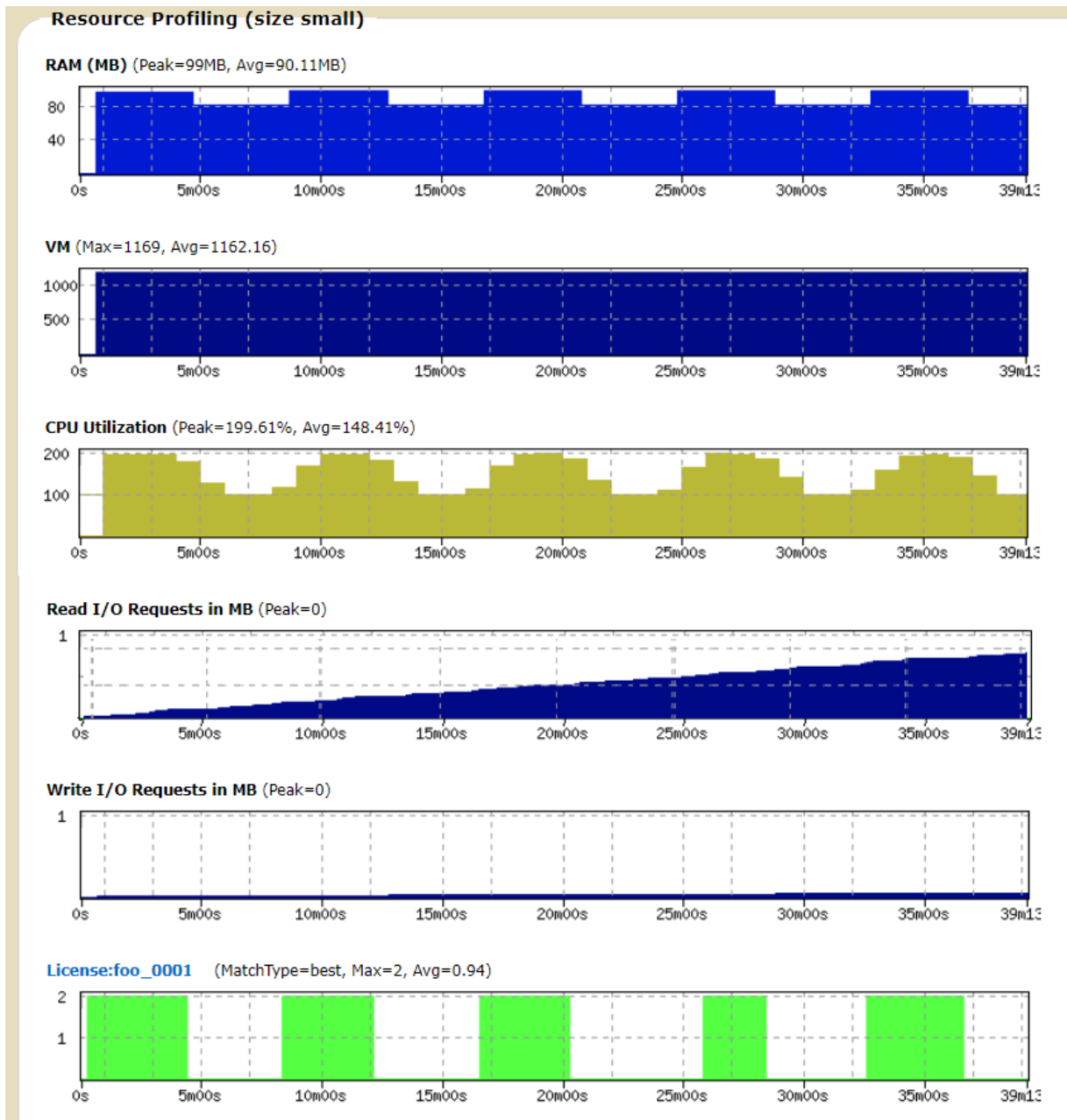


Figure 1:

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

```
% wx 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
...
```

List Jobs

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

The default is to show up to 20 jobs submitted by the user running `wx 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 Plus Administrator may have configured methods to mitigate this load, including caching.

wx list

```
wx: Usage Message

WX 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:
    % wx getfield $JOBID ...
        -- To get specific info about a job
    % wx cmd vovset count SETNAME ...
        -- To count jobs in sets by status
    % wx wait ...
        -- To block waiting for jobs to complete

USAGE:
```

```
% wx 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 WorkloadXelerator 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 WorkloadXelerator server in cases where list queries are scripted. By default, the cache expires after 30s.
- v -- Increase verbosity.

```
-h                -- Print brief usage.

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

Job Summary

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

```
wx: Usage Message

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

USAGE:
  % wx 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.
```

Examples

The following is an example of the output of `wx summary`:

```
% wx summary -all
Accelerator Plus 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`:

```
% wx summary -u john
Accelerator Plus 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'
```

Invoke the GUI

Job execution can be monitored with `wx gui`.

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

wx gui

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

```
wx: Usage Message

WX GUI:
  Show a grid view of the jobs in a specified set.
USAGE:
  % wx 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.
  -iopprofile <jobId> -- Show enhanced job profile.
  -taskers
  -fontsize <size>  -- Specify the normal font size. Default is 10.
                    Legal range is 3 to 36.
```



```
-title <title>      -- Choose title of X11 window.
```

EXAMPLES:

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

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

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

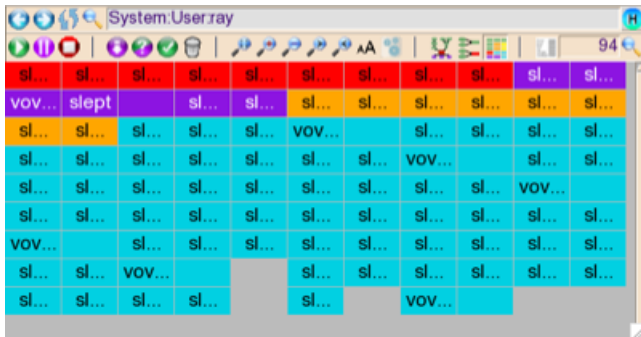


Figure 2:

```
% wx cmd vovconsole &
```

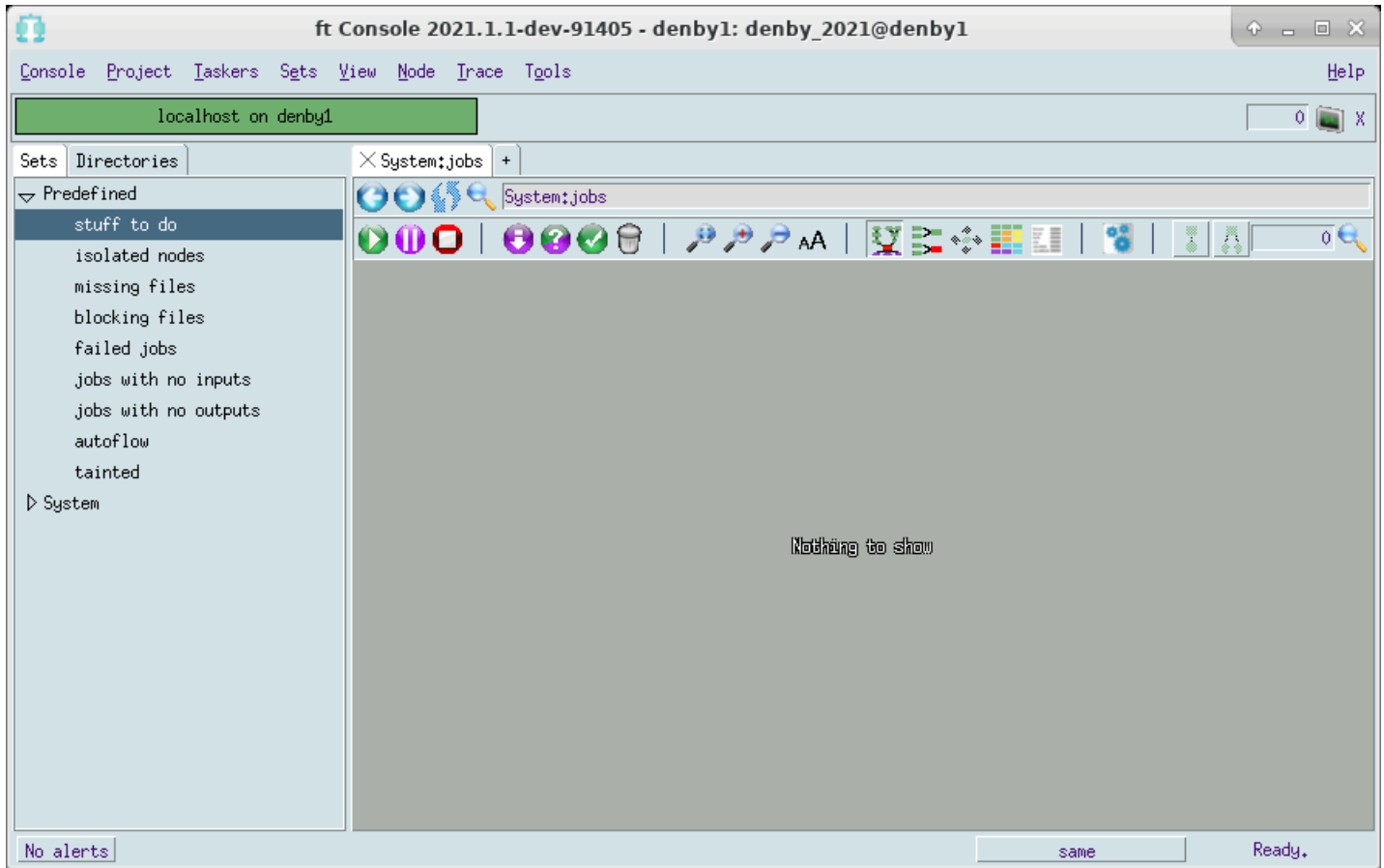






















Figure 3:

Icons

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

	Run/Rerun	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.
	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>ovvbuild</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 `wx hosts` shows the list of the hardware resources currently connected to an instance of Accelerator. These hardware resources are called "taskers" in Accelerator Plus.

```
wx: Usage Message
```

WX 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:

```
% wx 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:

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

An example is shown below:

```
% wx 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

% wx hosts -r
...
% wx 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
FULLINFO	FULLINFO	NA
GROUP	GROUP	NA
HB	NA	NA
HBPP	NA	NA
HEARTBEAT	HEARTBEAT	NA
HOST	HOST	NAME

nc hosts	vovselect from TASKERS	vovselect from HOSTS
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
PERCENT	PERCENT	NA
PERSISTENT	PERSISTENT	NA
PID	PID	NA
POWER	POWER	NA
RAM	RAM	NA
RAMFREE	RAMFREE	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
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
SLOTSTOTAL	SLOTSTOTAL	NA
STATSREJECTCORES	STATSREJECTCORES	NA
STATSREJECTOTHER	STATSREJECTOTHER	NA
STATSREJECTRAM	STATSREJECTRAM	NA
STATSREJECTRESERVED	STATSREJECTRESERVED	NA
STATSREJECTSLOTS	STATSREJECTSLOTS	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
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 Plus can be monitored with a dialog.

The dialog is invoked with:

```
% wx 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	Tasks	Tasker HW	Tasker Resources	Who	Running Jobs	Running Commands	Running Details	Resources	Queued Jobs	Queue Buckets	FairShare
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:

Manage Jobs

Accelerator Plus 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 `wx wait` command.

```
% wx wait [options] jobId ...
```

wx wait

```
wx: Usage Message

WX WAIT:
    Wait for a specific job or list of jobs to complete,
    fail, or be forgotten

USAGE:
    % wx 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:
% wx wait 22345 22356
% wx wait -dir .
% wx wait -subdirs .
% wx wait -select "command~spice"
% wx wait -set "myset"
% wx wait -callback ./myScript -set myset
% wx wait -jobinfo -set myset
% wx wait -p -set myset
% wx wait -poll 5000 -set myset
% wx 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 Plus administrator.

wx stop

wx: Usage Message

WX 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:

```
% wx 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:

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

```
% wx 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:

```
% wx 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 `wx 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.

wx forget

```
wx: Usage Message
```

```
WX 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:

```
% wx 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:

```
% wx forget -n  
% wx forget -age 1h  
% wx forget -mine -dir .  
% wx forget -allusers -dir .  
% wx forget -set MyExperiment  
% wx forget -set MyExperiment -forcerunning
```

Clean Up Log Files

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

wx clean

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

```
wx: Usage Message
```

WX 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:

```
% wx 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:

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

Examples

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

Comments

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

```
% wx clean -R
```

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

```
% wx 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:

```
% wx clean -deep 10
```


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

```
% wx clean -P 1d
```

Debug Jobs without Running Accelerator Plus

On occasion, jobs that run successfully outside of Accelerator Plus fail when run through Accelerator Plus. 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 `wx debug` can show you the steps that Accelerator Plus takes to run the job.

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

wx debug

```
wx: Usage Message
```

```
WX DEBUG:
```

```
If a job appears to behave differently when executed by WX  
than when it runs without using WX, 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 WX.
```

```
For example, if you find a job runs fine without WX, but fails  
in WX, 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:
```

```
% wx 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 Plus.

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 Plus 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
```

LSF Emulation

This document is intended for those who have been using the Platform LSF batch system and are now moving to using Accelerator Plus on top of the LSF batch 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 minimize the impact of transitioning workloads from using LSF directly and instead using the Accelerator Plus hierarchical scheduler. Ideally, the use of these scripts, will enable the redirection of workloads from using an LSF cluster directly to one using an Accelerator Plus scheduler in front of the same LSF cluster. While the scripts cover the common usage of LSF, higher performance and a richer set of scheduling directives will be achieved with using the native Accelerator Plus command set.

The emulated commands include the following:

- `bhist`
- `bhosts`
- `bjobs`
- `bkill`
- `bmgroup`
- `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
```

Configure Resource Mapping

The resources used by LSF, expressed by the `-R` option in `bsub`, are significantly different from the resource maps in Accelerator.

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® PollEx™ ©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 [55](#)

A

Accelerator Plus User Guide overview [4](#)

access to help [6](#)

ADE, Cadence [60](#)

allocate resources, FairShare [24](#)

B

bhist, emulated [60](#)

bhosts, emulated [60](#)

bjobs, emulated [60](#)

bkill, emulated [60](#)

bsub, emulated [60](#)

C

c-shell, TCSH user setup [12](#)

caching, nc list [37](#)

choosing the FairShare group [24](#)

clean up log files [55](#)

command line interface [11](#)

command line, run jobs [16](#)

command, wx [8](#)

command, wxmgr [9](#)

control the environment [19, 21](#)

CPU progress and run status indicators [34](#)

CPUPROGRESS, percentage of CPU time [34](#)

CPUTIME, total CPU time accumulated [34](#)

D

debug jobs example [58](#)

debug jobs without running Accelerator [58](#)

deep, clean option [55](#)

E

emulation, LSF [60](#)

environment control [19, 21](#)

example, submit a single job [17](#)

example, submit multiple jobs [18](#)

F

find jobclasses [23](#)

forget jobs [54](#)

G

get detailed information about a job [32](#)

H

help, Accelerator [6](#)

I

icons [42](#)

interactive jobs [27](#)

interactive jobs restrictions and consequences [30](#)

invoke the GUI [40](#)

J

job arrays [18](#)

job profiling [35](#)

job running overview [14](#)

job runtime - monitoring and profiling [34](#)

job status [33](#)

job submission arguments [24](#)

job summary [39](#)

jobclass, submit [23](#)

jobs, show current information [40](#)

L

listing jobs [37](#)

log file, interactive job [27](#)

log files [21](#)

LSF

 emulation [60](#)

LSF emulation [60](#), [60](#)

LSF, platform [60](#)

M

manage jobs [51](#)

metrics, scheduler [40](#)

modify running jobs [27](#)

modify scheduled jobs [29](#)

monitor job, RAM, CPU and children [34](#)

monitor workload [32](#)

monitoring jobs, taskers and resources [49](#)

multiple jobs, submit [18](#)

N

nc forget [54](#)

nc list [37](#)

nc stop [52](#)

nc wait [51](#)

nozap, clean option [55](#)

O

online help [6](#)

P

PDF, access [6](#)

post-condition [21](#)

pre-command and post-command job condition [21](#)

pre-condition [21](#)

pre-pending and appending arguments in a job submission [24](#)

priority [25](#)

Q

queue selection [16](#)

quick start [8](#)

R

run jobs with CLI commands [16](#)

running interactive jobs [27](#)

RUNSTATUS field values [34](#)

S

scheduled jobs [15](#)

scheduler metrics [40](#)

select a named environment [21](#)

setting up the user shell [12](#)

showing the hosts/taskers [44](#)

single job submit example [17](#)

status of jobs [33](#)

stop jobs [52](#)

stuck job, troubleshoot with LASTCUPROGRESS [34](#)

submission of jobs with pre-condition or post-condition [21](#)

submit jobs to Accelerator Plus [16](#)

submit jobs using jobclasses [23](#)

submit jobs with CLI commands [17](#)

submitted job information display [14](#)

T

troubleshoot stuck jobs [34](#)

U

use Accelerator help [6](#)

use jobclasses [22](#)

use vovselect for querying [45](#)

user setup: bourne shell, k-shell, z-shell, bash [12](#)

user setup: c-shell, TCSH [12](#)

user setup: Windows command shell [12](#)

user shell setup [12](#)

using snapshot with named environment [21](#)

V

verify your setup [13](#)

vnc_logs, cleaning [55](#)

vov [12](#)

VOV [60](#)

VOV_ENV [19](#)

VOV_JOB_DESC [35](#)

VOV_JOBINDEX [18](#)

VOV_LIMIT_cputime [52](#)

VOV_LOG_LSFEMUL [60](#)

VOV_STDOUT_SPEC [6](#)

VOV_STOP_SIGNAL_DELAY [52](#)

VOV_STOP_SIGNALS [52](#)

vovarch [12](#)

VOVARCH [19](#)

vovbrowser [6](#)

vovbuild [6](#), [42](#)

vovconsole [40](#)

VOVDIR [12](#), [35](#)

vovdoc [6](#), [6](#)

vovenvutils [19](#)

vovfsgroup [24](#)

vovid [6](#)

vovinit [12](#)

vovlsfemulib [60](#)

vovmemtime [52](#)

VovMessage [60](#)

VovPrintUsage [60](#)

vovrc [12](#)

vovselect [44](#)

vovserver [6](#), [58](#), [58](#)
vovsetupuser [12](#)
vovsh [60](#)
vovshow [24](#), [52](#)
vovtasker [34](#), [52](#), [58](#), [58](#)
vovtaskers [44](#)
vovversion [12](#)
vovwxd [4](#)
vsz, cleaning [55](#)
vtk_prop_get [60](#)
vtk_prop_set [60](#)
vtk_resourcemap_set [25](#)

W

waiting, stopping, cleaning, debugging jobs [51](#)
what happens when jobs are running [14](#)
wx clean [55](#)
wx command [8](#)
wx forget [54](#)
wx gui [40](#)
wx wait [51](#)
wxmgr command [9](#)