



# ALTAIR

ONLY FORWARD

Altair Accelerator 2024.1.0

Tutorials

# Contents

<b>Altair Accelerator User Tutorials</b> .....	4
Use Accelerator Help.....	5
Enable CLI Access on UNIX.....	7
Troubleshooting the UNIX Setup.....	9
Enable CLI Access on Windows.....	10
Verify Context Is Working.....	11
Run Basic Jobs.....	13
Get Summary Information.....	15
Run Jobs with Various Options.....	18
Submit Multiple Jobs at Once: -f <file>.....	18
Use Environments: -e <env>.....	18
Use of Resources: -r <res1 res2 ...>.....	19
Wait for Jobs.....	19
Specify Name of logfile: -l <logfile>.....	20
Job Control.....	21
Rerun Jobs.....	23
Get Detailed Information about a Job.....	24
Monitor Jobs, Taskers and Resources.....	25
Invoke the GUI.....	27
Use the Web Browser.....	29
Troubleshooting.....	30
<b>Altair Accelerator Administrator Tutorials</b> .....	31
Start a Test Queue.....	32
Start/Stop Accelerator.....	35
Browser-based Setup.....	37
Configure Policy - FairShare and Other Parameters.....	38
Server Configuration Parameters.....	39
Advanced Policy Configuration.....	40
Configure Resources.....	42
Configure Security.....	44
Configure Taskers.....	46
Configure an Environment.....	51
Logical Names (Equivalences).....	56
Resource Management.....	59
Altair Monitor-Basic Setup.....	59
Configure and Manage Monitor-basic.....	60
vtk_flexlm_monitor Procedures.....	61
Resource Throttling.....	61

Upgrade Accelerator.....	62
<b>Legal Notices</b> .....	65
Intellectual Property Rights Notice.....	66
Technical Support.....	70
<b>Index</b> .....	72

In the following User Tutorials, you will experiment with Accelerator on most topics that a user would be most interested in, including submitting jobs, tracking job information, analyzing and solving common problems, etc.

This chapter covers the following:

- [Use Accelerator Help](#) (p. 5)
- [Enable CLI Access on UNIX](#) (p. 7)
- [Enable CLI Access on Windows](#) (p. 10)
- [Run Basic Jobs](#) (p. 13)
- [Get Summary Information](#) (p. 15)
- [Run Jobs with Various Options](#) (p. 18)
- [Job Control](#) (p. 21)
- [Rerun Jobs](#) (p. 23)
- [Get Detailed Information about a Job](#) (p. 24)
- [Monitor Jobs, Taskers and Resources](#) (p. 25)
- [Invoke the GUI](#) (p. 27)
- [Use the Web Browser](#) (p. 29)
- [Troubleshooting](#) (p. 30)

## Use Accelerator Help

Accelerator documentation is available in HTML and PDF format.

### Access the Help when Accelerator is Running

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

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

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

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

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

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

### Access the Help when Accelerator is not Running

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

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

### Access the Help PDF Files

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

### Access the Help via the Command Line

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

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

### Access the Help via the vovshow Command

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

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

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

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

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

## Enable CLI Access on UNIX

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

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

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

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

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

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

- `vovrc.csh`
- `vovrc.sh`

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

Shell	Instructions
C-shell, tcsh	Add the following line to your <code>.cshrc</code> csh login script file: <pre>source /&lt;install_path&gt;/&lt;version&gt;/&lt;platform&gt;/etc/ vovrc.csh</pre>
sh, ksh, bash, zsh	Add the following line to your <code>.profile</code> shell login script file: <pre>./&lt;install_path&gt;/&lt;version&gt;/&lt;platform&gt;/etc/ vovrc.sh</pre>

## Verify Access to Altair Accelerator Products

After making the changes to source the helper files, and then logging in, you can check that needed environment variables are set properly by looking at environment variables to note that the `PATH` value contains a reference to the folders in the release

which hold programs, and to note that VOVDIR and VOVARCH are set. VOVDIR should contain the path to the installation. VOVARCH should contain the name that matches the machine type it is running on.

1. At the command prompt, enter the following:

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

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

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

```
% vovarch
```

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

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

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

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

or

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

## Enable Altair Accelerator for Non-Interactive Shells

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

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

Early exit from non-interactive shell login CSH script:

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



```
# Below are actions needed by interactive scripts.  
:
```

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

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

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

Source helper file before exit in login CSH script:

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

## Enable the Shell to Communicate With a Running Product Server

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

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

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

## Troubleshooting the UNIX Setup

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

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

```
% csh -c vovarch
```


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

## Enable CLI Access on Windows

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

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

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

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

### Method 1: Use Windows Explorer to Set Command Line Environment

1. Using Windows Explorer, navigate to the Altair Accelerator installation directory.
2. Enter the `win64/startup` folder and double-click the `vovcmd.bat` script to run it.  
This will open a command prompt with the proper environment settings for the Altair Accelerator and scripts to work.

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

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

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

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

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

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

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

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

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

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

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

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

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

rem -- Set Windows time from server on local network
rem -- Put this last; it may fail if lacking time set privilege
net time \\timehost /set /y
```

## Verify Context Is Working

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

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

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

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



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

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

## Enable the Command Prompt to Communicate With a Running Product Server

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

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

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

## Run Basic Jobs

Submitting jobs to Accelerator is quite simple: just use `nc run` followed by the command you would use without Accelerator.

### Run a Sleep Job


```
% cd # Go to your home directory
% nc run sleep 10
Resources= linux
Env = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command = vw vwrap sleep 10
Logfile = vnc_logs/20021230/103409.13784
JobId = 04194422
nc: message: Scheduled jobs: 1 Total estimated time: 0s
```

### Run Sleep Job with `-r` Option to Override Default Resources

By default, `nc run` takes the architecture of the machine from which the job is submitted as the job's resource. This can be overridden with `-r` option when needed.

This job requires no resource.

```
% nc run -r "" -- sleep 10
Resources=
Env = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command = vw vwrap sleep 10
Logfile = vnc_logs/20021230/103653.13794
JobId = 04194459
nc: message: Scheduled jobs: 1 Total estimated time: 0s
```

 **Note:** Since the `-r` option accepts multiple arguments, you need to terminate the resource list explicitly with another option, or `--` if no other options are needed.

This job requires `spice_license`.

```
% nc run -r spice_license -- sleep 10
Resources= spice_license
Env = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command = vw vwrap sleep 10
Logfile = vnc_logs/20021230/103948.13803
JobId = 04194497
nc: message: Scheduled jobs: 1 Total estimated time: 0s
```

### Run Job with `-e` Option to Override Default Environment

By default, `nc run` takes a snapshot of the environment as the job's environment and uses this to run your job. You can override this with `-e` option to use a named environment.

```
% nc run -e BASE sleep 10
Resources= linux
Env = BASE
Command = vw vwrap sleep 10
Logfile = vnc_logs/20021230/122737.14946
JobId = 04195924
```

```
nc: message: Scheduled jobs: 1          Total estimated time: 0s
```

Use the command `vel` to list all the available environments:

```
% vel
vel: message: Environment directories:
1 /remote/release/VOV/7.0u3/linux64/local/environments
1 . tcl BASE          UNIX utilities, X windows, and VOV.
1 . tcl D             Define variables: Usage: ves "+D(VAR1=value1,...)"
1 * tcl DEFAULT      Just a name for whatever you already have.
1 . tcl HSIM          Fake Nassda hsim env for Virage testing
1 . csh SYNOPSIS     Synopsys tools
```

Scripts that implement named environments may be written in `csh`, `sh`, or `Tcl` syntax. Cross-platform environments between Windows and UNIX/Linux must be written in `Tcl`. You can use named environments from the command line using the command `ves`.

## Get Summary Information

nc summary, nc list and nc info *jobId* will be the three commands that you use frequently to get information about the jobs managed by Accelerator.

### Get Summary of All My Jobs: nc summary

```
% nc summary
NC Summary For User dextrin
TOTAL JOBS      11      Duration: 23s
  Done          1
  Idle          0
  Queued        5
  Running       2
  Failed        3

  JOBS  GROUP  TOOL      WAITING FOR...
    1   alpha  sleep     'aa'
    3   alpha  sleep     'linux'
```

In the above example, the command displays summary information for user dextrin. There are a total of 11 jobs, 1 of which is Done, 2 Running, 3 Failed, etc. In the bottom part, it shows the summary for jobs that are queued, that is, there is 1 sleep job queued because it is waiting for resource 'aa', and there are 3 sleep jobs waiting for resource 'linux'.

To get an idea of what is going on in the whole Accelerator system, the following shows all the jobs and what the jobqueue buckets are waiting for. Note that Accelerator subcommands may be abbreviated.

```
% nc sum -a -b
```

### Get a List of My Jobs: nc list

This command, without any option, displays the last 20 of your jobs in the format of "jobId status command".

```
% nc list
04193437 Done      sleep 1
04193911 Done      sleep 5
04193913 Failed    sleep 10
04193917 Failed    sleep 20
04193919 Idle      sleep 60
04193937 Running   sleep 10
04193943 Queued    sleep 56
```

### Status Meaning

In Accelerator, each job is assigned an "Accelerator Computing Status" which is defined as follows:

Status	Color *	Explanation
Queued	Cyan	The job is scheduled to be executed.
Running	Orange	The job is currently executing
Done	Green	The job ran successfully

Status	Color *	Explanation
Failed	Red	The job ran and failed.
Idle	BlueViolet	The job needs to be run, but it is not scheduled.

\*(Colors may look different on some systems)

### Some Options about nc list

The `nc list` command has several useful options. They include:

Get detailed usage of this command:

```
% nc list -h
```

List all jobs, including others' jobs:

```
% nc list -a
04193437 Done    alpha    dextrin  rhino    sleep 1
04193898 Done    users   integ    rhino    sleep 2
04193939 Done    alpha    dextrin  rhino    sleep 15
04193941 Done    alpha    dextrin  rhino    sleep 20
04193943 Done    alpha    dextrin  rhino    sleep 60
```

Control output format, show job Id the (first) tool of job only:

```
% nc list -O "@ID@ @TOOL@"
04193437 sleep
04193898 sleep
04193939 sleep
04193941 sleep
04193943 sleep
```

In the above example, we use "fields", i.e., ID and TOOL, surrounded by two "@" signs, to format strings.

### Get Detailed Information About a Job: nc info jobId

Without options, this command displays the basic information about the job, like user, group, directory, command, environment, queue time, etc.

```
% nc info 04193937
Id,User,Group    04193937,dextrin,alpha
Environment      D(VOV_ENV_SOURCE=vnc_logs/envdextrin36362.env)
Directory        ${HOMES}/dextrin
Command          sleep 10
Status           Done
Host             rhino
QueueTime        1s
Duration         10s
Age              20m37s
AutoForget       1
```



With option -l, this command shows the contents of the log file.

```
% nc info -l 04193937
Log file is: '${HOMES}/dexin/vnc_logs/20021230/095337.13512.3'
vwrap: message: Start date: Mon Dec 30 09:53:38 PST 2002
vwrap: message: On host: rhino
vwrap: message: Sourcing environment vnc_logs/envdexin36362.env
vwrap: message: Running: 'sleep 10'
vwrap: message: Exit status: 0
vwrap: message: End    date : Mon Dec 30 09:53:48 PST 2002
```

## Run Jobs with Various Options

In this section, we will exercise some options in `nc run` command.

To get the detailed usage, use the following command:

```
% nc run -h
```

### Submit Multiple Jobs at Once: `-f <file>`

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 -f commandFile
```

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

### Use Environments: `-e <env>`


1. Get a list of all available environments:

```
% vel  
vel: message: Environment directories:  
1 /<install_path>/local/environments  
1 . tcl BASE          UNIX utilities, X windows, and Flowtracer.  
1 . tcl D             Define vars: Usage: ves "+D(VAR1=value1,...)"  
1 . tcl DEFAULT      Just a name for whatever you already have.  
1 . csh SPICE        The Analog simulator SPICE3.  
1 . csh JAVA         JAVA Development Environment (1.2.2)
```

2. Use proper environment(s) to submit jobs, for example:

```
% nc run -e BASE sleep 10  
% nc run -e BASE+SPICE spice chip.spi
```

## Use of Resources: -r <res1 res2 ...>

 **Note:** Since this option accepts multiple arguments, you need to terminate the resource list explicitly with another option, or "--" if no other option is needed.

1. Submit a job that only runs on Linux machine:

```
% nc run -r os=linux64 -- sleep 10
```

2. Submit a job that requires resources "License:spicy" and "hspice":

```
% nc run -r License:spicy -- sleep 10
```

## Wait for Jobs

Waiting for jobs is especially useful for scripts. By default, `nc run` returns immediately after you submit a job. This allows you to submit multiple jobs at once. There could be times when you want to run jobs in sequence, in which case option `-w` is very useful. With this option, `nc run` waits for the job(s) to finish ( Done or Failed ).

```
% nc run -w sleep 10
Resources= linux
Env       = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command  = vw vwrap sleep 10
Logfile  = vnc_logs/20021231/111314.21781
JobId    = 04211346
vnc: message: Scheduled jobs: 1          Total estimated time: 0s
```

## Wait for Jobs and Show Log File of the last job: -wl

With this option, `nc run` waits for the job(s) and shows the log file of the last job.

```
% nc run -wl date
Resources= linux
Env       = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command  = vw vwrap date
Logfile  = vnc_logs/20021231/111530.21819
JobId    = 04211392
nc: message: Scheduled jobs: 1          Total estimated time: 0s
<<<STARTING ON rhino>>>
Tue Dec 31 11:15:31 PST 2002
<<<END OF LOG>>>
<<<EXIT STATUS 0>>>
```

Please only use `-wl` for jobs where you are actively monitoring the output. Such jobs listen to the vovservers event stream and are called 'notify clients'. They are more resource-intensive than plain batch jobs.


## Wait for Jobs Using `nc wait`

You can also wait for jobs using the command `nc wait`. We will cover that in next tutorial "Job Control".

## Specify Name of logfile: **-l <logfile>**

The default logfile name has the form of `./vnc_logs/date/time`. You can explicitly specify the logfile of the jobs you submitted by this option, for example:

```
% nc run -l /home/john/logs/log1.log sleep 10
```

 **Warning:** Conflicts may occur if same log file is used for multiple jobs. New users are not recommended to use this option.

## Job Control

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

### Wait for Jobs

Besides option `-w` and `-wl` with `nc run` command, you can also wait for job(s) to finish (Done or Failed) after they are submitted using the command `nc wait`. Here are some examples:

*Get usage help*

```
% nc wait -h
```

*Wait for a job*

```
% nc run sleep 10
Resources= linux
Env       = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command  = vw vwrap sleep 10
Logfile   = vnc_logs/20021231/140024.23307
JobId     = 04213283
nc: message: Scheduled jobs: 1          Total estimated time: 0s

% nc wait 04213283
```

*Wait for all jobs in current directory*

```
% nc wait -dir .
nc: message: Job 04193913 is already FAILED
nc: message: Job 04193915 is already FAILED
nc: message: Job 04193917 is already FAILED
nc: message: Job 04193919 is not scheduled
nc: message: Job 04194308 is not scheduled
nc: message: Job 04211259 is already FAILED
nc: message: Job 04211268 is not scheduled
nc: message: Job 04213283 is already VALID
nc: message: Job 04213297 is already VALID
nc: message: Exiting with status 2 (Failed jobs)
```

*Wait for all jobs using tool spice*

```
% nc wait -select "tool==spice"
```

In the above example we use "Selection Rules" to perform a "wait" on those jobs that satisfy that rule. This could be useful for other commands as well, including `nc list`, `nc forget`, etc. Refer to the *Altair Accelerator User Guide* for more information.

### Forget Jobs

The Accelerator server remembers the jobs you submitted for some configurable time. You can explicitly forget them by `nc forget` command, which will delete all job information from the server database.

*Get a list of jobs*

```
% nc list
04146420 Done      sleep 1      #in the form of "jobId status command"
```

```
04146425 Done    sleep 5
04146427 Running sleep 10
04146429 Running sleep 15
04146431 Queued  sleep 20
04146433 Queued  sleep 60
```

### Forget some of them

```
% nc forget 04146420 04146425
nc: message: Forgetting 2 jobs
```

### List jobs again (notice those two are gone)

```
% nc list
04146427 Done    sleep 10
04146429 Done    sleep 15
04146431 Done    sleep 20
04146433 Done    sleep 60
```

### Forget all my jobs

```
% nc forget -mine
nc: message: Forgetting 4 jobs
```

## Stop Jobs

A job can be stopped when it is either Running or Queued. Stopping a job does not forget it from the server database. "Running" jobs will exit, and "Queued" jobs will be dequeued when you stop them.

### Stop some jobs

```
% nc run sleep 60
Resources= linux
Env       = D(VOV_ENV_SOURCE=vnc_logs/envdixin36362.env)
Command   = vw vwrap sleep 60
Logfile   = vnc_logs/20021231/140803.23386
JobId     = 04213393
nc: message: Scheduled jobs: 1          Total estimated time: 0s

nc stop 04213393
nc: message: Stopping RETRACING job 04213393
```

### Stop all my jobs

```
% nc stop -mine
nc: message: Stopping RETRACING job 04146627
nc: message: Stopping RETRACING job 04146629
nc: message: De-queuing  job 04146631
nc: message: De-queuing  job 04146633
```

## Rerun Jobs

The `nc rerun` command initiates the scheduling and execution of jobs that are already in the server database. By default, only the jobs that are Idle or Queued are affected by this command. If you want to force the rerunning of jobs that are either Done or Failed, use the option `-F`.

*Rerun a "Done" job won't do anything*

```
% nc rerun 04146622
nc: message: Not rerun: 04146622
```

*Force rerunning a "Done" job*

```
% nc rerun -F 04146622
nc: message: Job 04146622 is already VALID.
nc: message: Scheduled jobs: 1          Total estimated time: 1s
```

*Rerun "Idle" jobs*

```
% nc rerun 04146631 04146633
nc: message: Scheduled jobs: 1          Total estimated time: 0s
nc: message: Scheduled jobs: 1          Total estimated time: 0s
```

## Get Detailed Information about a Job

The command `nc` displays information about a job.

### Get Detailed Information About a Job

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

```
nc: Usage Message

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

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

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

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

### Examples:

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



## Monitor Jobs, Taskers and Resources

The activity of Accelerator can be monitored with a dialog.

The dialog is invoked with:

```
% nc monitor
```

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

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

TaskerGroups	Taskers	Tasker HW	Tasker Resources	Who	Running Jobs	Running Commands	Running Details	Resources	Queued Jobs	Queue Buckets	FairShare
Type:Name	Total	InUse	Rsrvd	ooQ	Available	%Util.	MapsTo				
1	License:AL002_HMPoll	unlim	0	0	0	unlim					
2	Limit:user0265_justo	1	0	0	0	1	99,12%				
3	License:AL002_HFSoli	100000	0	0	0	100000	0,00%				
4	License:AL003_HMPSD_	unlim	0	0	0	unlim					
5	Limit:user0418_justo	1	0	0	0	1	99,79%				
6	Limit:user0354_justo	1	0	0	0	1	97,64%				
7	Limit:user0100_justo	1	0	0	0	1	98,94%				
8	License:AL003_HMPSD_	unlim	0	0	0	unlim					
9	License:SolverNode	100000	0	0	0	100000	0,00%				
10	Limit:user1312_justo	1	0	0	0	1	98,16%				
11	License:AL003_HMPano	unlim	0	0	0	unlim					
12	License:AL003_SIMLAB	100000	0	0	0	100000	0,00%				
13	License:AL001_HMActi	unlim	0	0	0	unlim					

Figure 1:

## Invoke the GUI

Job execution can be monitored with `nc gui`.

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

### nc gui

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

```
nc: Usage Message

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

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

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

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

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

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

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

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

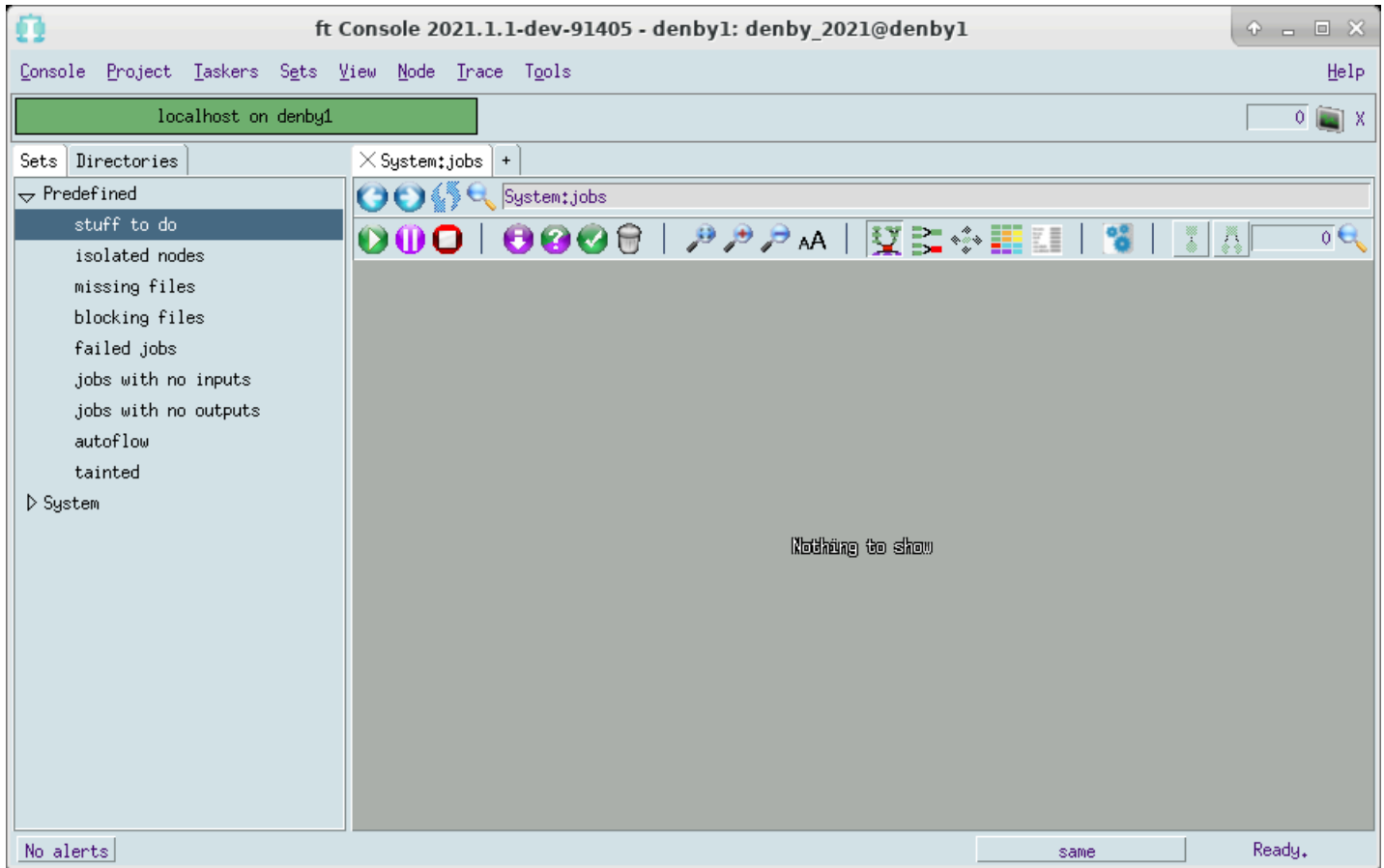


Figure 2: GUI that opens after entering nc cmd vovconsole &

# Use the Web Browser

## Find the URL for Accelerator

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

## Use the Browser Interface

Enter the URL found above into your browser's location box. You will need to login unless your administrator has turned off authentication.

`http://your_host:your_port_number/project` is the home page of Accelerator. From this page, you can easily navigate to many other useful pages. Here we list some monitor pages that are similar to the ones from the GUI monitors.

<b>Tasker page</b>	From the home page, click <b>Taskers</b>
<b>Workload pages (job queue and running jobs):</b>	From the home page, click <b>Workload</b> For running jobs, under Workload, click <b>Running jobs</b>
<b>Resources page</b>	From the home page, click <b>Resources</b>
<b>FairShare page</b>	From the home page, under Workload, click <b>Fairshare</b>

Among the many useful pages, two report pages may be especially helpful.

<b>Job queue report</b>	From the home page, click <b>Workload &gt; Job Queue Reports</b>
<b>Resources reports</b>	From the home page, click <b>Resources &gt; Resource Reports</b>

## Troubleshooting

This section covers typical problems that come up in Accelerator. The command `nc summary` will be useful here, as it tells you how many jobs are failed, queued, or idle, and what the queued jobs are waiting for. For example:

```
% nc summary
NC Summary For User bkkring
TOTAL JOBS          0      Duration: 0s
Done                0
Idle                0
Queued              0
Running             0
Failed              0
```

### My job won't start!

Often, your job won't start because it is waiting for a resource, usually a license, CPU, or memory.

To diagnose this, use the command:

```
nc info jobid
```

A job will only start if all resources requested by the job are available. If any resource is missing, the job will not start. You can look at the resources of all available vovtaskers to see if there is any that can run the job with the command:

```
nc host
```

Additionally, a vovtaskers must either be `READY` or `WRKNG` (have a free job slot) to accept jobs. Any other condition will prevent the vovtaskers from taking the job.

### My job failed!

You can find the reason for job failure with the command:

```
nc info jobid
```

Some common failure conditions include:

- The job failure has nothing to do with Accelerator. Run the job without Accelerator to verify this.
- The job command doesn't exist, possibly because of a typo.
- You are using a wrong, nonexistent, or incomplete environment with the `-e`. In this case, `nc info jobid` will tell you that it cannot switch to the environment.
- You have failed to specify (or specified the wrong) architecture or memory usage. This can be done with the `-r` option. For example, `-r linux64` for Linux 64 bit or `-r RAM/2000` for 2GB of ram.

# Altair Accelerator Administrator Tutorials

---

In the following tutorial, you will experiment with most issues that an Accelerator administrator will need to address, including starting/stopping the server, configuring FairShare, resources, taskers and environments.

This chapter covers the following:

- [Start a Test Queue](#) (p. 32)
- [Start/Stop Accelerator](#) (p. 35)
- [Browser-based Setup](#) (p. 37)
- [Configure Policy - FairShare and Other Parameters](#) (p. 38)
- [Advanced Policy Configuration](#) (p. 40)
- [Configure Resources](#) (p. 42)
- [Configure Security](#) (p. 44)
- [Configure Taskers](#) (p. 46)
- [Configure an Environment](#) (p. 51)
- [Logical Names \(Equivalences\)](#) (p. 56)
- [Resource Management](#) (p. 59)
- [Upgrade Accelerator](#) (p. 62)

You will start a test queue to make the experiment non-disruptive to the default Accelerator queue.

## Start a Test Queue

At a given site, a single queue is recommended, such as a single Accelerator setup, by default called "vnc". Such a setup is called a *cluster* by other systems. The scheduler in Accelerator does not have queues in the sense used by other batch systems.

In this tutorial, you will start a temporary Accelerator queue for our testing. This allows you to experiment with most of the administration tasks without disturbing your production Accelerator queue.

## Find the Server Working Directory

The Accelerator configuration directory for the default queue named "vnc" is in the product hierarchy. The server working directory for this queue is:

```
$VOVDIR/../../../../vnc/vnc.swd
```

This directory contains the server configuration files and server/tasker logs, etc.

## Start a Queue

The command to start a queue is `ncmgr start`. By default, this command will start the default queue in the directory of `$VOVDIR/../../../../vnc`.

1. Get usage of this command:

```
vncmgr: Usage Message

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



```

configured to enable REST API v3 interface,
to enable the dashboard web UI page,
and to enable SSL. A value of 0 directs
VovServer not to open a web interface port.
Specify port number, port number list (colon
separated) or port mode. Modes are:
automatic - hash queue name into port number,
do not start if port is
unavailable. The default queue
name 'vnc' hashes to web port
6271.
any - hash queue name into port number, try
additional ports in increments of 1
until an available one is found. The
default queue name 'vnc' hashes to
beginning web port 9695.
Default: Any
-webprovider <provider> Specify the provider for
HTTP and HTTPS protocols.
This must be either "internal" or "nginx".
Default: "internal"
-roport <port|mode> Specify read-only guest access web interface
port. A value of 0 disables this interface,
requiring all web interface users to log in.
Default: 0
-q, -queue <name> Name for queue (default is $NC_QUEUE if set,
and otherwise vnc).

-dir <dir> Directory of the server
(default $VOVDIR/../../vnc).
-dbhost <host> Host for database.
-dbroot <path> Path on database host for database files.
-dbport <port> Port of the database to listen for
connections.
-v Increase verbosity.
EXAMPLES:
% ncmgr start -port 6271
% ncmgr start -port 6271:6272:6273:any -force
% ncmgr start -q bigqueue -dir /remote/queues

```

2. Start the test queue in the home directory.
3. Name your queue vncdixin (make sure you pick a name that does not conflict with any existing queue). The first three letters of the queue name should be formed by prefixing your login with vnc  
For example, if your login is 'danny', use the name 'vncdanny' for your queue.

```

+ start a shell on the machine where your NC vovserver should run
% cd # Go to your home directory
% mkdir ncadmin # Create a directory for our testing queue
% ncmgr start -dir ncadmin -queue vncdixin
message: Checking the license...
message: ... the license is good.
message: Starting NC
message: with name vncdixin
message: on host alpaca
message: in directory /home/dexin/ncadmin
message: as user dexin
Do you want to proceed? (yes/[no]) > yes
message: Updating config file '/remote/release/VOV/2013.09/linux64/local/
vncConfig/vncdixin.tcl' message: Waiting for server to be ready ...
message: Sanity check...

```

```
message: NC vncdexin@alpaca is ready.
```



**Note:** Make sure the directory `$VOVDIR/local/vncConfig` is writable to you, because a config file needs to be written in that directory to start a new queue. The Altair Accelerator installer should have set this.

## Use a Specific Queue

Now that you have started a new queue, you have at least two Accelerator queues in your system. There are two ways to use a particular queue.

1. Use `-q` or `-queue` option:

```
# Submit a job to queue "vncdexin"
% nc -q vncdexin run sleep 60

# List my jobs in queue "vncdexin"
% nc -q vncdexin list

# Get info about queue "vncdexin"
% ncmgr info -queue vncdexin

# Reset all taskers for queue "vncdexin"
% ncmgr reset -queue vncdexin -taskers
```

2. Alternatively, you can set the environment variable `NC_QUEUE`:

You can also set an environment variable `NC_QUEUE` to the name of queue you want. Then you can execute all your `nc` and `ncmgr` commands in the context of that queue without having to use `-q` or `-queue` options.

```
% setenv NC_QUEUE vncdexin

# Now submit a job to queue "vncdexin"
% nc run sleep 61


# etc.
```

This way, you can easily switch between all the queues you have.

## Start/Stop Accelerator

The commands involved in this tutorial are `ncmgr info`, `ncmgr start` and `ncmgr stop`.

You should use `ncmgr`, especially when starting Accelerator, because it checks for and applies any daemon and DB schema changes when moving to a new version.

 **Note:** Please remember to use the shell where you set the `NC_QUEUE` environment variable to the name of your training tutorial queue.

## View Accelerator Status

Use command `ncmgr info` to get a summary information of all taskers and current running jobs.

```
% ncmgr info
ncmgr: message: NC vncdixin@alpaca
1 001 alpaca 1/0 146198 Unlim. IDLE
2 002 cheetah 1/0 235294 Unlim. IDLE
3 004 bison 1/1 99206 Unlim FULL
   2s:      vw vwrap sleep 60 > vnc long
4 000 pluto 1/0 75312 Unlim. IDLE
5 003 rhino 1/0 181818 Unlim. IDLE
```

## Start Accelerator

1. Execute `ncmgr` to see if Accelerator is already running.

```
% ncmgr start
ncmgr: USER ERROR: NC vncdixin@alpaca already running.
```

2. If not already running, start Accelerator. In production use, it is important to check the number of file descriptors available to `vovserver`.

```
% ncmgr start
ncmgr: message: Checking the license...
ncmgr: message: ... the license is good.
ncmgr: message: Starting NC
ncmgr: message: with name          vncdixin
ncmgr: message: on host            alpaca
ncmgr: message: in directory       /home/dexin/ftadmin
ncmgr: message: as user           dixin
ncmgr: message: with             1024 file descriptors
Do you want to proceed? (yes/[no]) > yes
ncmgr: message: Waiting for server to be ready ...
ncmgr: message: Sanity check...
ncmgr: message: NC vncdixin@alpaca is ready.
```

## Stop Accelerator

Run the following:

```
% ncmgr stop
ncmgr: message: Checking if NC vncdextrin@alpaca is running...
ncmgr: message:
You are about to stop NC vncdextrin@alpaca in directory /home/dextrin/ftadmin
    Would you like to proceed (yes/[no]) ? > yes
ncmgr: message: Stopping taskers and server ...
ncmgr: message: NC vncdextrin@alpaca has been stopped.
```

## Restart Your Accelerator Queue

You should restart your queue so that it will be running for later exercises.

Run the following:

```
% ncmgr start
ncmgr: message: Checking the license...
ncmgr: message: Checking the license...
ncmgr: message: ... the license is good.
ncmgr: message: Starting NC
ncmgr: message:     with name         vncdextrin
ncmgr: message:     on host           alpaca
ncmgr: message:     in directory      /home/dextrin/ftadmin
ncmgr: message:     as user           dextrin
    Do you want to proceed? (yes/[no]) > yes
ncmgr: message: Waiting for server to be ready ...
ncmgr: message: Sanity check...
ncmgr: message: NC vncdextrin@alpaca is ready.
```

## Browser-based Setup

Accelerator provides a simplified user-friendly browser-based setup, which is especially useful for new users. To use this browser-based setup page, please first make sure Accelerator is running.

### Find the URL

First find the Accelerator URL, using `vovbrowser` or `vsi`:

```
% nc cmd vovbrowser
http://yourhost:6295/project
% nc cmd vsi
(more detailed output, including the Accelerator vovserver URL)
```

The setup script is available at the URL `/cgi/setup.cgi` (for example, in this case, it is `http://alpaca:6295/cgi/setup.cgi`).

### Setup Using the Web Page

1. Follow the instructions on the web page to finish the basic setup.
2. On the Taskers setup page, try to add at least one tasker for each type (Server, Workstation, Offhours).
3. View the tasker statuses at URL `/taskers`.
4. Try out some test jobs.



**Note:** You need a working remote-shell setup to start non-local vovtaskers for this step.

## Configure Policy - FairShare and Other Parameters

Accelerator has a time-windowing multi-level FairShare mechanism. This allocates CPU cycles to the fsgroups at each level according to the assigned weights of each group until the leaf nodes of the FairShare tree are reached. Each node of the FairShare tree may have a separate time window. The word 'fsgroup' is used as an abbreviation here.

To configure FairShare, use the `vovfsgroup` command.

Access to FairShare groups is controlled by ACLs (Access Control Lists), which means you can configure them so only designated users can submit jobs in an fsgroup.

### Locate Server Configuration Directory to Find `policy.tcl`

The test queue was start in `vncdextrin` in directory `~/ncadmin`. So the server configuration directory for this Accelerator queue is `~/ncadmin/vncdextrin.swd`. By default, the server configuration directory is `$VOVDIR/./././vnc/vnc.swd`.

If you forget, you can find it out by this command:

```
% nc cmd vovserverdir
/home/dextrin/ncadmin
```

In this case, you will find `policy.tcl` file at `/home/dextrin/ncadmin/vncdextrin.swd/policy.tcl`.

You will find other configuration files in the same directory, for example, `taskers.tcl`, `resources.tcl`.

### Configure `policy.tcl` for FairShare

In the following example, two groups, `production` and `regression`, are configured.

Execute the following:

```
# in policy.tcl
% vovfsgroup create /production -weight 400 -window 8h
% vovfsgroup create /regression -weight 100 -window 8h
```

Two groups are defined: `/production` and `/regression`. When there are jobs from both the `production` group and `regression` group, the target share ratio of CPUs will be 400:100.

### Save the New Configuration

Changes made by the `vovfsgroup` command are only in the `vovserver`'s memory until the next save, and may be lost if the `vovserver` is restarted before then.

Save the configuration to a file so it can be reloaded.

```
% nc cmd vovfsgroup genconfig myfsconfig.tcl
```

## Test the New Fairshare Configuration

1. You use `-g` option in command `nc run` to submit a job from a particular group.

```
% nc run -g regression -f listOfJobs
% nc run -g production -f listOfJobs
```

2. Use the Monitors GUI or browser page to monitor the dynamic changes of FairShare.
  - Use the monitors: use command `nc monitor` to bring up the monitors and click on the **FairShare** tab.
  - Use the browser: use command `nc cmd vovbrowser` to find the Accelerator URL, then find the FairShare page at URL `/cgi/fairshare.cgi`.

## Server Configuration Parameters

Certain parameters of the Accelerator `vovserver` are also configurable by means of entries in the `policy.tcl` file. The [Server Configuration](#) page describes these in detail.)

Here are some examples:

```
# This is part of the policy.tcl file.
set config(maxQueueLength)      8000
set config(httpSecure)          1
set config(saveToDiskPeriod)    2h;
set config(autoLogout)          1h;    # Logout from browser interface

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

## Advanced Policy Configuration

Besides the vovserver configuration file `policy.tcl`, the behavior of job submission to Accelerator can also be controlled by the file `$VOVDIR/local/vnc_policy.tcl`, which is used to define the following procedures:

<code>VncPolicyDefaultResources</code>	The default resources required by a job.
<code>VncPolicyValidateResources</code>	Make sure that the resource list for a job obeys any number of rules.
<code>VncPolicyDefaultPriority { user }</code>	Assign the default priority to a job based on the user.
<code>VncPolicyMaxPriority { user priority }</code>	Limit the priority based on the maximum allowed to the user.

In this tutorial, you will configure `VncPolicyDefaultResources` and `VncPolicyValidateResources`.

### Configure Default Resources

By default, Accelerator takes machine architecture as the resource of the job that you submit from a particular machine. This is controlled by the default setting of `VncPolicyDefaultResources`:

```
proc VncPolicyDefaultResources {} {  
    global env  
    return "$env(VOVARCH)"  
}
```

To change this behavior, you can create or edit the file `$VOVDIR/local/vnc_policy.tcl`, and add or edit the procedure that follows. This procedure will set the default resources to be the architecture and 50mb of memory. If you have more than one Accelerator setup, you can place the `vnc_policy.tcl` file in the `.swd` and it will apply only to that one.

```
proc VncPolicyDefaultResources {} {  
    global env  
    return "$env(VOVARCH) RAM/50"  
}
```

### Enforce Job Resource Rules

You can also enforce some rules of job resources by overriding the procedure `VncPolicyValidateResources`. Here is the default behavior:

```
proc VncPolicyValidateResources { resList } {  
    return $resList  
}
```

But this process does nothing. Try the process described below:



To enforce a rule so that all jobs require a minimum RAM of 512 MB, create or edit the file `$VOVDIR/local/vnc_policy.tcl` and add or edit `VncPolicyValidateResources` procedure as follows:

```
proc VncPolicyValidateResources { resList } {  
  #  
  # This policy adds a minimum RAM requirement  
  # for all submitted jobs.  
  #  
  if [regexp "RAM/" $resList] {  
    # Already a RAM constaints.  
  } else {  
    lappend resList "RAM/512"  
  }  
  return $resList  
}
```

## Configure Resources

Accelerator includes a sophisticated subsystem for the management of computing resources, which allows the design team to take into account all sorts of constraints regarding hardware and software resources, as well as site policy constraints. This mechanism is based on:

- The resources required by jobs
- The resources offered by taskers
- The ResourceMap, as described in the file `resources.tcl`

Next, you will configure the resource map in `resources.tcl`.

## Find and View the resources.tcl File

You can find this file in the server configuration directory. For default vnc queue, it is `$VOVDIR/./././vnc/vnc.swd`. For the test queue, it is in directory `~/ncadmin/vncdexin.swd/`.

```
% cd ~/ncadmin/vncdexin.swd
% vi resources.tcl ; # Use the editor of your choice
```

```
# ... here we only show part of this file ...
vtk_resourcemap_set PRIORITY_LOW      1
vtk_resourcemap_set PRIORITY_NORMAL  10
vtk_resourcemap_set PRIORITY_HIGH     20
vtk_resourcemap_set PRIORITY_TOP      UNLIMITED
```

With above default configuration, there will be at most 1 low priority job running at any time, 10 for normal, 20 for high, and any number of top priority jobs could be running.

1. Run a simple test to verify that the above information is correct:

```
% nc run -f $VOVDIR/training/vnc/cmdlist.unix
```

2. You can use the Monitors window to monitor the "Running Jobs" and "Resources" to see the running jobs and resources usage. You can also use browser to get similar information from "Running Jobs" page and "Resources" page.

## Configuration Examples

### Example 1

For example, if you decide that no more than 4 normal priority jobs should be running at any time, you can edit `resources.tcl` and modify the value for normal priority to 4.

```
# ... here we only show part of this file ...
vtk_resourcemap_set PRIORITY_LOW      1

# Now we change this value to 4
vtk_resourcemap_set PRIORITY_NORMAL  4
```

```
vtk_resourcemap_set PRIORITY_HIGH 20
vtk_resourcemap_set PRIORITY_TOP UNLIMITED
```

Save your change and do a reread and test the new configuration:

```
% nc cmd vovproject reread
% nc run -f $VOVDIR/training/vnc/cmdlist.unix
```

## Example 2

You can configure your resources similarly. For example, you have 10 calibre license, you can configure this by adding the following line in `resources.tcl`:

```
# In resources.tcl
vtk_resourcemap_set calibre_license 10
```

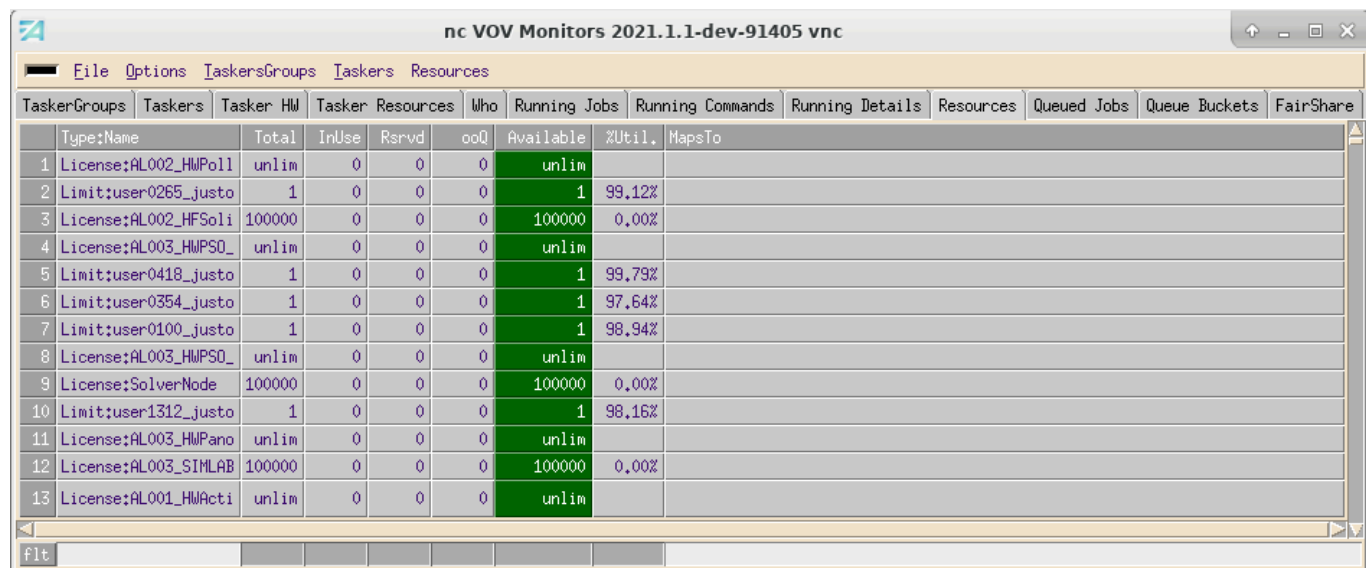
You can also associate a resource to other resource(s). For example, you have 4 licenses of spice that are only available on Linux. You can configure this by add the following line in `resources.tcl`:

```
# In resources.tcl
vtk_resourcemap_set hspice_license 4 linux
```

## Resource Monitor

To monitor resource activity, call a GUI monitor with the command:

```
nc mon
```



The screenshot shows a window titled "nc VOV Monitors 2021.1.1-dev-91405 vnc". The window contains a menu bar with "File", "Options", "TaskersGroups", "Taskers", and "Resources". Below the menu bar is a table with columns: "TaskerGroups", "Taskers", "Tasker HW", "Tasker Resources", "Who", "Running Jobs", "Running Commands", "Running Details", "Resources", "Queued Jobs", "Queue Buckets", and "FairShare". The table lists 13 resources with their status and usage percentages.

TaskerGroups	Taskers	Tasker HW	Tasker Resources	Who	Running Jobs	Running Commands	Running Details	Resources	Queued Jobs	Queue Buckets	FairShare
1	License:AL002_HWPoll	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_HWPSO_	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_HWPSO_	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_HWPano	unlim	0	0	0	unlim					
12	License:AL003_SIMLAB	100000	0	0	0	100000	0.00%				
13	License:AL001_HWActi	unlim	0	0	0	unlim					

Figure 3:

## Configure Security

Accelerator has 4 privilege levels: READONLY, USER, LEADER, ADMIN.

For detailed information about security, please refer to [Security](#).

## Locate the Security Configuration File: security.tcl

This file is in the server configuration directory, default \$VOVDIR/../../vnc/vnc.swd/security.tcl and in our test setup, that is ~/ncadmin/vncdexin.swd/security.tcl.

## Security Configuration Examples

### Least Restrictive Security

The least restrictive security grants everybody full access from any host. This should not be used in production.

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

Alternatively, a VovUserGroup may be utilized, to assign individuals in a group the ADMIN privilege.

```
# Members of mygroup are administrators from all hosts (+).  
vtk_security -group mygroup ADMIN +
```

### Most Restrictive Security

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

### Typical Case

The following example shows a typical security file, in which different privileges are granted to different users. Also notice the use of variables and VovUserGroups in this example.

In the example, mary is an administrator for any host, and dan is an administrator only for reno and milano. The user pat is a LEADER for her machine elko, and fred has USER privileges for 4 machines listed in the variable \$allhosts. Members of the VovUserGroup "operators" have ADMIN rights on \$allHosts.

```
set servers          { reno milano }  
set allhostsset     { reno milano elko tahoe}  
  
vtk_security mary    ADMIN    +  
vtk_security john    ADMIN    tahoe  
vtk_security dan     ADMIN    $servers  
vtk_security pat     LEADER   elko  
vtk_security fred    USER    $allhosts
```

```
vtk_security -group operators ADMIN $allHosts
```

## Configure Taskers

The file `taskers.tcl` describes the vovtaskers for Accelerator.

### vovtasker Configuration

You can find the file `taskers.tcl` in server configuration directory, default at `$VOVDIR/../../vnc/vnc.swd/taskers.tcl`, and in our test case at `~/ncadmin/vncdexin.swd/taskers.tcl`.

This file is based on two commands (Tcl procedures), `vtk_tasker_define` and `vtk_tasker_set_default`:

```
# Set default behavior of vovtaskers
vtk_tasker_set_default [options]

# Define a vovtasker
vtk_tasker_define hostname [options]
```

If you set some options with `vtk_tasker_set_default` command, all the following `vtk_tasker_define` commands will use those options implicitly, and options set explicitly in `vtk_tasker_define` overwrite those set in `vtk_tasker_set_default`.

For example, to declare 3 vovtaskers on the hosts `apple`, `orange`, and `pear`, use:

```
# In taskers.tcl
vtk_tasker_define apple
vtk_tasker_define orange
vtk_tasker_define pear
```

or some equivalent code:

```
# In taskers.tcl
foreach host {apple orange pear} {
    vtk_tasker_define $host
}
```

To understand the use of `vtk_tasker_set_default`, define 3 vovtaskers as follows:

```
vtk_tasker_define apple -resources "@STD@ big_memory" -CPUS 2
vtk_tasker_define orange -resources "@STD@ big_memory" -CPUS 2
vtk_tasker_define pear -resources "@STD@ big_memory" -CPUS 4
```

`vtk_tasker_set_default` does the following:


```
vtk_tasker_set_defaults -resources "@STD@ big_memory" -CPUS 2
vtk_tasker_define apple
vtk_tasker_define orange
vtk_tasker_define pear -CPUS 4
```

## Tasker Configuration Examples

The default `taskers.tcl` provides pretty good examples of configuring vovtaskers. Most of the time, you just need to plug in some host names in brackets to define vovtaskers. For example, here is one piece of code in `taskers.tcl`.

```
# ADD THE NAMES OF THE COMPUTE SERVERS TO THE FOLLOWING LIST
set mainComputeServers {}

foreach host $mainComputeServers {
    vtk_tasker_define $host -resources "VovResources::Standard @RAMTOTAL@ @SWAPFREE@"
}
```

 **Note:** `-resources "VovResources::Standard"` is equivalent to `-resources"@STD@"`

To add some "Server" class vovtaskers, you just need to add the names of those hosts into the list `mainComputerServers`, like the following:

```
set mainComputeServers { apple orange pear }
```

You can certainly add or modify the vovtasker definition as you want (subject to license restriction). For example, you have many dual CPU machines, and you would like to make the *maxload* of these vovtasker machines bigger, say, equal to 1.5 times of the CPUs, for example 3.0. Then you can do this:

```
set myDualCpuServers {}

foreach host $myDualCpuServers {
    vtk_tasker_define $host -maxload 3.0 -resources "VovResources::Standard
@RAMTOTAL@ @SWAPFREE@"
}
```

## Add Workstation/Offhours vovtaskers

```
# Add a Workstation vovtasker that will only start to accept
# job after 10 minutes of ilde and will only accept jobs
# with expected duration no longer than 5 minutes
vtk_tasker_define ftcsun44 -resources "VovResources::Workstation -minIdle 10m -
maxtime 5m @RAMTOTAL@"

# Add a Offhours vovtasker that's only available from 7pm to 6am
# every weekday and on weekends
vtk_tasker_define ftcsun66 -resources "VovResources::Offhours"
```

## Start Newly Defined vovtaskers

Like other configurations, Accelerator will pick up the changes in `taskers.tcl` when the server is stopped and restarted. This is often too disruptive and not favorable, especially when there are jobs running in Accelerator. Here are some other ways to apply your changes:

1. To start the vovtaskers that you just defined, use command:

```
% ncmgr reset -taskers
```

2. To start vovtaskers that you just modified, you must stop and start them.

## Start/Stop vovtaskers (Advanced)

You can start/stop vovtaskers from the GUI and the browser.

The following is an advanced command of FlowTracer vovtaskermgr and how to run any command in the context of Accelerator using `nc cmd`.

```
# List all vovtaskers defined in taskers.tcl
# Also checks the taskers.tcl file for syntax errors
% nc cmd vovtaskermgr list

# Start all vovtaskers defined in taskers.tcl
% nc cmd vovtaskermgr start

# Start some vovtasker(s)
% nc cmd vovtaskermgr start tasker1 tasker2

# Stop all vovtaskers
% nc cmd vovtaskermgr stop

# Stop some vovtasker(s)
% nc cmd vovtaskermgr stop tasker1 tasker2

# Show detail information of all vovtaskers
% nc cmd vovtaskermgr show

# Get usage of all vovtaskermgr commands
% nc cmd vovtaskermgr
```

## Start a vovtasker from Command Line (Advanced)

You can also start a vovtasker on the fly from the command line using the `vovtasker` binary. This can sometimes be handy, for example, for debugging. This is the command that Accelerator uses to start the vovtaskers after it reads the `taskers.tcl` configuration file.

1. To start a vovtasker on a particular host, you need to go to that host and use command `nc cmd vovtasker` with appropriate options, which are similar to the options of `vtk_tasker_define`.
2. Try the following examples and monitor the vovtaskers using the Monitor GUI or browser Tasker page.
  - Get usage of this command:

```
% nc cmd vovtasker
usage: vovtasker [-A startupLogFile] [-a name] [-b capabilities] [-B]
               [-c coefficient] [-C cpus] [-d] [-D integer] [-e reserveExpr]
               [-E] [-f tclfile] [-F <file>]
               [-g taskergroup] [-G group] [-h host] [-H HEALTHCHECKFLAGS]
               [-I 0|1] [-I tclfile] [-j] [-k d|n|v] [-l rootofDailyLogFile]
```



```

[-L <loadSensor>] [-m <integer>][-M max_load] [-n <integer>]
[-N] [-o local resource] [-p project]
[-P <double>] [-r resources] [-R resources] [-s] [-S
resources] [-t timeout] [-T capacity/max_capacity]
[-U updateInterval] [-v number] [-V ncName@ncHost[:port]]
[-w WX properties] [-z <timeSpec>] [-Z <timeSpec>] [-g name]
[-u name]

-A: The name of the startup log file
-a: Name this tasker. The name may contain only letters, numbers,
dash(-) and underscore(_), or the expressions @HOST@ and @PID@ that get
expanded on the fly
-b: Comma-separated list of capabilities, case insensitive:
    symbolic: FULL NC LM
    normal: PROCINFO NC LM
    short: P N X R
-B: Show BPS tasker objects. Default to not show.
-c: Tasker coefficient (positive, default 1.0)
-C: Number of CPU's in this machine (automatic on win64). Use 0
to specify default value

...OMITTED...

```

- Start a normal vovtasker (with all default settings):

```

% nc cmd vovtasker -N
vovtasker Jan 10 13:44:42
Copyright © 1995-2020, Altair Engineering Linux/7.1 Jan 10 2019 10:00:59
vnc@alpaca
vovtasker Jan 10 13:44:42 Test 1: INTEGER OPS W= 1.00 Reps= 500 T= 5.00ms
vovtasker Jan 10 13:44:42 Test 1: DOUBLE OPS W= 1.00 Reps= 25 T= 10.00ms
vovtasker Jan 10 13:44:42 Test 1: CHAR OPS W= 0.10 Reps= 10 T= 10.00ms
vovtasker Jan 10 13:44:42 ---- Weighted time: 16.00ms
vovtasker Jan 10 13:44:42 Test 2: INTEGER OPS W= 1.00 Reps= 500 T= 5.00ms
vovtasker Jan 10 13:44:42 Test 2: DOUBLE OPS W= 1.00 Reps= 25 T= 10.00ms
vovtasker Jan 10 13:44:42 Test 2: CHAR OPS W= 0.10 Reps= 10 T= 20.00ms
vovtasker Jan 10 13:44:42 ---- Weighted time: 17.00ms
vovtasker Jan 10 13:44:42 Test 3: INTEGER OPS W= 1.00 Reps= 500 T= 5.00ms
vovtasker Jan 10 13:44:42 Test 3: DOUBLE OPS W= 1.00 Reps= 25 T= 10.00ms
vovtasker Jan 10 13:44:42 Test 3: CHAR OPS W= 0.10 Reps= 10 T= 20.00ms
vovtasker Jan 10 13:44:42 ---- Weighted time: 17.00ms
vovtasker Jan 10 13:44:42 Best weighted time: 16.00ms

```

- Start a vovtasker with name "myvovtasker", max load 4.0, and offers standard resources "@STD@" and resource "special\_license":

```

% nc cmd vovtasker -a myvovtasker -M 4.0 -r "@STD@ special_license"
vovtasker Jan 10 13:51:00
Copyright © 1995-2020, Altair Engineering Linux/7.1 Jan 10 2019 10:00:59
vnc@alpaca
vovtasker Jan 10 13:51:00 Test 1: INTEGER OPS W= 1.00 Reps= 500 T= 5.00ms
vovtasker Jan 10 13:51:00 Test 1: DOUBLE OPS W= 1.00 Reps= 25 T= 5.00ms
vovtasker Jan 10 13:51:00 Test 1: CHAR OPS W= 0.10 Reps= 10 T=
20.00ms
vovtasker Jan 10 13:51:00 ---- Weighted time: 12.00ms
vovtasker Jan 10 13:51:00 Test 2: INTEGER OPS W= 1.00 Reps= 500 T= 5.00ms
vovtasker Jan 10 13:51:00 Test 2: DOUBLE OPS W= 1.00 Reps= 25 T= 5.00ms
vovtasker Jan 10 13:51:00 Test 2: CHAR OPS W= 0.10 Reps= 10 T=
20.00ms
vovtasker Jan 10 13:51:00 ---- Weighted time: 12.00ms
vovtasker Jan 10 13:51:00 Test 3: INTEGER OPS W= 1.00 Reps= 500 T= 5.00ms
vovtasker Jan 10 13:51:00 Test 3: DOUBLE OPS W= 1.00 Reps= 25 T= 5.00ms

```

```
vovtasker Jan 10 13:51:00 Test 3: CHAR OPS    W= 0.10 Reps= 10    T=  
20.00ms  
vovtasker Jan 10 13:51:00 ---- Weighted time: 12.00ms
```

## Configure an Environment

VOV includes a sophisticated subsystem to manage environments. In Accelerator, an environment is a named collection of environment variables and their values. Many tools expect certain environment variables to be set for correct operation, and the variable `PATH` is used by most UNIX shells to locate executable programs.

A VOV environment definition consists of three scripts defined in either Tcl, C-shell, or Bourne-shell syntax. Environments that will be used on both UNIX and Windows must be written in Tcl. The names of these scripts must follow a naming convention. For example, an environment named `ENV` would use these three scripts:

- `ENV.start.tcl`
- `ENV.end.tcl`
- `ENV.doc`

The 'start' script is used when entering the environment, the 'end' script is used when leaving the environment, and the 'doc' script contains a one-line summary of the environment's purpose.

VOV provides commands for using environments from the command line:

<code>vel</code>	List available environments
<code>vesENV</code>	Switch to environment ENV
<code>vep</code>	Change prompt to show environment name
<code>veprestore</code>	Return to original prompt

## Find and View Example Environment Scripts

It is interesting and instructive to examine some of the environment definitions which are shipped with VOV. One of the most important is the `BASE` environment.

The most common place for environment scripts is `$VOVDIR/local/environments`, but you can also specify additional directories where VOV will search. We will discuss this in more detail later in the lab.

The `BASE` definition is in the 'local' directory of the VOV software installation, at `$VOVDIR/local/environments/BASE.start.tcl`

```
% pushd $VOVDIR/local/environments
% view BASE.start.tcl

# --
# -- Completely reset the environment to bare bones.
# -- We assume that VOVDIR and VOVARCH are set.
# --

set VOVDIR $env(VOVDIR)

if { $::tcl_platform(platform) eq "windows" } {
#
```

```

# --
#
nt_preprocess_env
# set vovenv(debug) 1 set WINDIR $env(WINDIR)
setenv PATH "$WINDIR\\system32;$WINDIR"

set MKSDIR "c:/mksdemo"
if [file exists $MKSDIR] {
    setenv ROOTDIR $MKSDIR
    vovenv PATH ";" APPEND $MKSDIR/mksnt
}
#
# Depending on the installation, the binaries can be in
# different directories.
#
foreach b { bin bin-0 bin-g bat local/bat } {
    set p "$VOVDIR/$b"
    #
    # Eliminate double // as in w://bat
    #
    regsub -nocase {^([a-z])://} $p {\1:/} p

    if [file exists $p] {
        vovenv PATH ";" APPEND $p
    }
}

set version [exec voversion]
vovenv PATH ";" PREPEND c:/temp/vov/execache$version

# vovenv PATH ";" PREPEND c:/temp/vov/execache

} else {
    #
    # -- Unix BASE environment.
    #

    setenv PATH ""
    foreach b { bin bin-0 bin-g } {
        if [file exists $VOVDIR/$b] {
            vovenv PATH : APPEND $VOVDIR/$b
            break
        }
    }
    vovenv PATH : APPEND $VOVDIR/scripts
    vovenv PATH : APPEND $VOVDIR/local/scripts

    setenv MANPATH ""
    foreach mp [list /usr/man /usr/share/man /usr/local/man /usr/openwin/man $VOVDIR/man]
    {
        if [file isdirectory $mp] {
            vovenv MANPATH : APPEND $mp
        }
    }

    setenv LD_LIBRARY_PATH ""
    foreach lp [list /lib /usr/lib /usr/local/lib /usr/openwin/lib $VOVDIR/lib] {
        if [file isdirectory $lp] {
            vovenv LD_LIBRARY_PATH : APPEND $lp
        }
    }

    foreach xapp { /usr/openwin/lib/app-defaults /usr/lib/X11/app-defaults } {

```

```
if [file isdirectory $xapp ] {
    vovenv XFILESEARCHPATH : APPEND "$xapp/%N"
}
}

foreach p {
    /bin /usr/ucb /usr/bin /usr/local/bin
    /usr/X11/bin
    /usr/dt/bin
    /usr/openwin/bin /usr/bin/X11
} {
    if [file isdirectory $p] {
        vovenv PATH : APPEND $p
    }
}
}
```

The BASE environment includes just the regular system commands and VOV. This is an example of a complex environment setup script; most of the ones you will need to define will be much simpler.

A simple test to verify that:

```
% vep
```

Notice that the prompt has changed to show that you are in the DEFAULT environment. This is the environment which is defined by your regular system setup files, for example, `.cshrc`.

```
% ves BASE
% printenv PATH
```

Notice that the PATH environment variable is now (probably) much shorter, and contains the VOV software directories.

```
% veprestore
```

Notice that the prompt has been changed back to the original.

 **Note:** This only changes the prompt. Your shell still has whatever environment was set most recently using `vep`.

## Add Additional Environment Directories

When you are testing and developing environments, or when you want to have project-specific ones which are not put in the system wide directory at `$VOVDIR/local/environments`, you can use the environment variable `VOV_ENV_DIR` to specify additional directories.

You will use this capability in the tutorial, so that you don't need to put our environment scripts in the system's shared environment area. You should have a `~/ncadmin` directory from the previous exercises, which contains the server working directory of your private Accelerator queue.

```
% setenv VOV_ENV_DIR ~/ncadmin/vnc-user-name.swd/environments
% pushd $VOV_ENV_DIR
```



**CAUTION:** The `enviroments` directory is generated by setup, but any user-made env-scripts placed in the directory will not be used unless `VOV_ENV_DIR` includes it.

## Configuration Examples

In this example, you will create an environment which adds your personal bin directory to the path. You will put the definitions in your queue's environment directory.

Use Tcl for the environment scripts because it is more powerful than C-shell, and platform-independent.

### Example 'start' script for JOHN environment

```
# add john bin directory to path
vovenv PATH : PREPEND /home/john/bin
```

This script adds the element `/home/john/bin` to the beginning of the `PATH`. There is also an `APPEND` operator which adds to the end of the `PATH`. If the exact element is already present in the path, it is left undisturbed. This avoids `PATH` overflow in C-shell.

### Example 'end' script for JOHN environment

```
# remove john bin directory from path
vovenv PATH : DELETE /home/john/bin
```

This script removes the element `/home/john/bin` from the `PATH`.

### Example 'doc' script for JOHN environment

```
prepend /home/john/bin to PATH
```

This file contains a one-line summary which is presented by the `vel` command.

Use the above examples to create similar files in your queue.

## Compose Good Environment Scripts

There are several characteristics of a 'good' environment setup.

#### **minimal**

A minimal environment only sets the variables needed to accomplish a specific purpose.

#### **composable**

A composable environment respects pre-existing values of variables, and adds to them, rather than overwriting them. For example, when modifying `LM_LICENSE_FILE`, use `PREPEND` and `APPEND` rather than simply setting it.

This allows you to use commands like `ves BASE+CADENCE+NASSDA` and have both Cadence and Nassda tools work.

**reversible**

A good environment definition uses the 'end' script to revert what it set in the 'start' script, so that the environment does not gradually become polluted with obsolete values.

**Environments are Cached on the vovtasker**

To provide very low job dispatch latency, Accelerator caches the eight most-recently used environments in the vovtasker process. This means that jobs which run in any of those environments may start immediately without sourcing the environment definition script. It also means that if you change an env-script, you must tell the vovtaskers by using the `vovtaskermgr refresh` subcommand.

**Summary**

- Most environments are defined by scripts stored in `$VOVDIR/local/environments`
- Environment scripts are in Tcl or C-shell syntax
- You can use the `VOV_ENV_DIR` environment variable to specify additional directories to search for environment scripts.
- Environments are cached on the vovtasker, and refreshed using `ncmgr reset -taskers`, or `nc cmd vovtaskermgr refresh`


## Logical Names (Equivalences)

Most sites set up their machines so that they all have uniform mountpoints and view of the file systems. A common scheme is to place all user home directories under a single parent directory, often called /home. The actual file system is mounted using automount, and the physical file system is located by an NIS map.

Unlike some other batch/network computing systems, Accelerator does not require that all the compute machines have a uniform set of mountpoints. Accelerator can use what are called logical names or equivalences, to locate the data.

The default `equiv.tcl` file which is shipped with the software defines a logical name for the path to VOVDIR.

The `equiv.tcl` file in some versions also defines a logical name HOMES for whatever the parent of your home directory is set in the environment variable HOME. The default one shipped with an early version did not, so we will show the steps to define a logical name using the browser-based setup.

 **Note:** When doing the exercises, remember to give these commands from a shell where you have set the environment variable NC\_QUEUE to the name of your training tutorial queue. Otherwise, you will need to use the `-queue` option.

## Define a Logical HOME Directory Name

1. Use the browser-based setup to view the logical names that are defined. First, find the URL of the queue's server, and connect a web browser to it.

```
% nc cmd vovbrowser --> http://host:port/project
```

```
% firefox http://host:port/cgi/setup.cgi
```

The initial page will have a link on the left called FileSystems.

2. Left-click on this link to display the logical names defined for the queue. You will see VOVDIR and VNCSWD.
3. Run a job in your home directory.

```
% cd  
% nc run sleep 10
```

This will often run, because the default is to require the architecture from which the job was submitted as a resource, so the job may run on your machine.

4. Now try the job, but request an architecture different from the machine on which you are working. This job will fail, because the directory on the remote machine does not have a logical name.

```
% nc run -r linux64 -e BASE pwd
```

```
Resources= linux64 Env = BASE  
Command = vw vwrap pwd
```



```
Logfile = vnc_logs/20030108/143125.27906 JobId = 00000021
vnc: message: Scheduled jobs: 1 Total estimated time: 0s
```

```
% nc list
00000011 Done sleep 10
00000021 Failed pwd
```

```
% nc info -l 21
```

```
Log file is: '${VNCSWD}/vncjohn.swd/vnc_logs/20030108/143125.27906'
vnc: message: File ${VNCSWD}/vncjohn.swd/vnc_logs/20030108/143125.27906 does not
  exist (yet)
  (mapped to /usr2/home/john/ncadmin/vncjohn.swd/
  vnc_logs/20030108/143125.27906)
```

##### 5. Run the job in the /tmp directory.

/tmp exists on all UNIX machines, but is not usually exported to the network.

```
% cd /tmp
% nc run sleep 10
```

This fails, because /tmp is not a network path.

```
% nc run sleep 10

WARNING:
  The path to the current directory is not 'LOGICAL',
  i.e. it does not begin with a '$' sign.
EXPLANATION:
  The current directory
  /tmp
  may not be a valid directory path for all hosts in the network.
OPTIONS:
  (1) Continue.
      By choosing this option, you assert that the path
      is valid everywhere. If it is not, the job is likely to fail,
      because the remote host cannot reach the current directory.
      This option causes the creation of a flag file called .vnc which has
the
      purpose of avoiding the repetition of this question for this directory
and
      its subdirectories
  (2) Abort.
      Please ask your NetworkComputer administrator to change the equiv.tcl
file to
      define the rules that give a logical name to the current directory.
```

If you respond **2**, the job will not be submitted.

If you respond **1**, a file named .vnc will be created in the directory, and the job will be submitted. In the future, you will not receive the warning so long as the .vnc file exists.

The Altair Accelerator software internally uses logical names to refer to files.

## Advanced Commands

### 1. Resource the cache file.

```
ls ~/ncadmin/vncjohn.swd/equiv.caches  
cayman pluto  
cayman abbcanova@cayman DEFAULT+SUPPORT html/ftnctraining > cat !$/pluto
```

```
cat ~/ncadmin/vncjohn.swd/equiv.caches/pluto  
VOVDIR /remote/release/VOV/7.0u2/linux64  
VOVDIR /remote/release/VOV/7.0u2/linux64/../../common
```

### 2. Run the vovequiv command.

```
% vovequiv -p . # show the logical name for this directory
```

## Resource Management

This exercise will walk you through some of the more advanced and complex issues of resource management in Accelerator. You should already be familiar with Accelerator and basic resource management, covered in the [Configure Resources](#) tutorial.

With Accelerator, Altair Accelerator includes a simplified version of our Monitor product, called Monitor-basic, which interfaces between NC and licensing systems like FlexNet Publisher. Unlike the full LM product, LM-basic does not do history or denials.

### Review

The following topics are review from the [Configure Resources](#) tutorial:

- Find and view `resources.tcl` file
- The `vtk_resourcemap_set` procedure
- Static Resource (PRIORITY\_LOW) configuration example
- Configure resources map on the fly using `vtk_resourcemap_set`

Topics of this tutorial include:

- Monitor-basic setup
- `vtk_flexlm_monitor` procedure
- Browser-based setup
- Resource throttling

In this tutorial, we will start LM-basic, configure it, and use it to monitor licenses. There are two things necessary to monitor FlexNet Publisher features:

1. Configure and start the LM-basic vovserver.
2. Accept the default `vtk_flexlm_monitor_all`, or add specific `vtk_flexlm_monitor` statements to NC's `resources.tcl`

## Altair Monitor-Basic Setup

Altair Accelerator uses a two-layer interface to FlexNet Publisher. There is a daemon `vovresourced` which is started by the NC server whenever there are `vtk_flexlm_monitor` statements in the `resources.tcl` file. `vovresourced` gets license information from `licmcon` via HTTP, and uses it to maintain NC's License: `resources`.

In the case where many NC queues or FlowTracer projects are running, it greatly reduces the load on FlexNet Publisher and improves the consistency of the license information to run Monitor or Monitor-basic. Additionally, Monitor offers a browser interface which shows licenses in use and license utilization information. Altair Accelerator recommends that you run this whenever you need to monitor FlexNet Publisher licenses, even in cases where you only have one server running Accelerator.

## Configure and Manage Monitor-basic

Both Monitor-basic and full Monitor are managed by the `lmmgr` command. It is essential that you use this instead of other ways to start the LM vovserver, because it also takes care of any DB schema updates, auxiliary daemon and startup changes needed by newer versions.

You use the configuration file of `vovflexlmd` in Monitor-basic to specify which FlexNet Publisher licenses you want the daemon to monitor. You do not need to monitor all licenses. In this exercise, we will monitor one. In most cases it is easier to use the browser UI to configure which license servers are monitored.

1. Change to the working directory of the daemon and see whether it is already running.
2. If you see an `info.tcl` file, then check whether the process named there exists on that system.

```
% cd $VOVDIR/local/vovflexlmd; ls
```

Example `info.tcl` file:

```
# This file is created automatically by vovlmd: DO NOT EDIT
# If you touch this file, the process 17400 will exit.
set host      "jaguar"
set pid       17400
set port      5555
set cwd       "/remote/proj9/cadmgr/licmon/licmon.swd/vovlmd"
set version   "2.0"
set timestamp 1441146545; # Tue Sep 01 15:29:05 PDT 2019
```



**Note:** If this directory already contains a file named `config.tcl`, your site may already be running. In this case, check for the file `info.tcl`. This file will contain the host name, process ID, and port of the daemon. You can verify that it is running by pointing your web browser to the URL `http://host:port` using the port and host in the info file.

Even if the daemon is not running, we will use our own subdirectory to run our instance Monitor-basic, rather than use the default directory.

3. Create a directory to run your own Monitor-basic instance.

```
% mkdir ~/ncadmin/licmon
% cd ~/ncadmin/licmon
```

4. Start Monitor-basic, picking a port which is different from the regular one, if you are running on the Accelerator server host. The default TCP/IP port number is 5555. You can find the ports in use by `vovproject list -a -l`. For this example, we use port 5575.

```
% cd ~/ncadmin/licmon
% lmmgr start -name licmon<user> -dir . -port 5575
```

5. Prepare a configuration file with a text editor, which names one of the FlexNet Publisher license files at your site.

```
% vovproject enable licmon<user>
% cd `vovserverdir -p vovlmd`
% vi config.tcl
```

6. Edit the default configuration file `config.tcl`, and enter at least one license file to monitor, using the `add_LM_LICENSE_FILE` procedure. The license may be in any format acceptable as a `-c` parameter to `lmstat`, that is, a full pathname or in `port@host` notation. It is better to use `port@host` than file paths, so that NFS is not involved. Example `config.tcl` file.

```
# config.tcl -- vovflexlmd FLEXlm configuration
add_LM_LICENSE_FILE -tag CDNS /remote/vendors/cadence/license/license.dat
add_LM_LICENSE_FILE -tag SNPS /remote/vendors/synopsys/license/pluto.dat
add_LM_LICENSE_FILE -tag SUNW 1726@saturn
```

7. Verify that LM-basic is running and supplying license information. Examine the `info.tcl` file created by the `vovlmd` daemon, and point your web browser to the URL as described above.
8. Stop the LM-Basic `vovserver`.

Most of the Accelerator daemons may be stopped by touching the `info` file that they create. When it starts, the daemon records the timestamp of the file. Each cycle, the daemon examines the timestamp, and exits if it has changed.

For Monitor-basic, the auxiliary daemons like `vovlmd` will stop when the Monitor-basic `vovserver` stops.

## vtk\_flexlm\_monitor Procedures

Your Accelerator setup will get license in-use info from your Monitor or Monitor-basic setup via HTTP on Monitor's `/raw` interface. The default `resources.tcl` file for Accelerator includes the `vtk_flexlm_monitor_all` procedure. This converts every feature monitored by Monitor into an Accelerator resource. If the resource name is specified, then this name is the actual resource name used. If the resource name is not specified, the name defaults to `License:<feature>`.

If you do not wish to have a large number of resources, you can comment out the above and put calls to the `vtk_flexlm_monitor` procedure in your `resources.tcl` file to specify which FlexNet Publisher features Accelerator should monitor.

## Resource Throttling

In some cases, you may wish to restrict the number of a feature which are consumed by Accelerator jobs, to leave some for 'fast' or interactive use.

To accomplish this, you can use an auxiliary resource called `throttle_<resource>` to restrict the number of Accelerator jobs. For example, you have 15 licenses of `hsm`, and want to leave 1 free for jobs that bypass the Accelerator system.

You would create a resource 'throttle\_hsm' with a count of 14, by adding statements like this. Map `hsm_license` to `throttle_hsm`, so that when the throttle resource is exhausted, jobs that need `hsm` will queue. Often such resources are named `Limit:<something>`.

```
vtk_flexlm_monitor hsm hsm_license throttle_hsm
vtk_resourcemap_set throttle_hsm 14
```

## Upgrade Accelerator

There are two types of updates for Accelerator, patches and new versions. Patches replace specific files in an installation and save an archive of the changed files so you can revert. A new release is a complete set of files, usually installed in a directory that is a sibling of your current version.

The simplest method to change to a different server is to use the following steps (shutdown and restart on new version):

- Install new software
- Notify users that Accelerator will be shutdown
- At an idle time, shut down the Accelerator server with `ncmgr stop`
- Change startup files (`.vovrc`) to refer to the new software installation
- Start the server, running the new version, with `ncmgr start`
- Notify users that Accelerator is available again

In some cases, for example, it may be impractical to shut down the Accelerator server for even a short time:

- you have a very busy installation, and Accelerator is never idle
- you have jobs running and you do not want to lose them
- you want to keep the order of jobs in the queue. In such cases you can use one of the following methods.
  1. Switch `vovserver` and `vovtaskers` separately, using `ncmgr stop -freeze`.
  2. Start a new Accelerator queue on new version, sending new jobs to it, shutting down the current Accelerator queue after all jobs have been retired.

Here are the steps for the `ncmgr stop -freeze` method.

1. From a shell with your current Altair Accelerator version, stop `vovserver` with `ncmgr stop -freeze`. This instructs `vovtaskers` with jobs to wait for a new `vovserver` and reconnect to it.
2. `vovtaskers` with no jobs should exit right away, and ones with jobs will enter the SUSP state. Their names will change to indicate they are stopping (and to permit new `vovtaskers` to start with the regular names).
3. From a shell with the new Altair Accelerator version, start the `vovserver` using `ncmgr start`.
4. Check on the browser UI Admin page that `vovserver` is running the expected version.

Here are the steps for the overlapping queues method.

1. Create a new queue, running the new server software
2. Direct `nc run` to send new jobs to the new queue
3. After some time, all jobs in the old queue have completed
4. The old queue may be now be shut down

## Overlapping Queues

Please read these steps to the end and understand them before doing the procedure. This procedure is preliminary, and may need to be adapted to your Accelerator configuration.

1. Install and verify the new software version. The `vovcheck` command may be helpful.

2. Start a new queue, for example with name `vnc2`.

```
% ncmgr start -queue vnc2
```

3. Copy configuration files (`policy.tcl`, `resources.tcl`, `taskers.tcl` etc.) from the configuration directory of the old queue to that of the new queue in such a way that the two queues are functionally the same.
4. Get the new queue ready.

```
% ncmgr reset -queue vnc2 -full
```

5. In the `setup.tcl` file of the new queue, set the environment variables `NC_OLDQUEUE` and `NC_OLDVERSION` to point to the old queue.

```
# This is a fragment of the setup.tcl file for the new queue
# USE THESE FOR MIGRATION FROM AN OLD QUEUE.
setenv NC_OLDQUEUE
vnc setenv NC_OLDVERSION 5.4.7
```



**Note:** This goes in the `setup.tcl` file so that these environment variables will always be set in the context of `vnc2`.

6. Test the new queue. By setting the environment variable `NC_QUEUE` to `vnc2`, the Accelerator commands will use that as the default queue.

```
% setenv NC_QUEUE "vnc2"
% nc run sleep 10
% nc list
% nc mon
```



**Note:** You may use the `-queue` option as an alternative to setting `NC_QUEUE`. For example, you might use

```
% nc -queue vnc list -a      # Show all jobs in old queue.
% nc -queue vnc2 run sleep 10 # Submit a sleep job to new queue.
% nc -queue vnc2 list       # Show your jobs (only) in new
                             queue.
```

7. Once you are satisfied that the second queue is ready, make it the default queue. The default queue is always named `vnc`, and is where the newly-submitted jobs will be placed.

```
% cd $VOVDIR/local/vncConfig
% mv vnc.tcl vnc1.tcl; ln -s vnc2.tcl vnc.tcl
```



**Note:** In the example above, `vnc.tcl` is moved to `vnc1.tcl` to preserve its contents, in case you want to reactivate the original queue. The two commands are on the same line, separated by a semicolon, so that the file `vnc.tcl` will be unavailable for the shortest possible time. Check carefully!

Now all the newly-submitted jobs will go to `vnc2`, while commands such as `nc info`, `nc stop`, `nc forget`, etc. will be run on both queues.

8. Error messages are filtered out, because a job ID which is valid in the first queue will not be valid in the second, and conversely. To see the results with messages, query the queues separately by using the `-queue` option of Accelerator commands.
9. When all jobs in the old queue have been retired, you can shut down the old queue.



# Legal Notices

# Intellectual Property Rights Notice

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

Copyright © 1986-2023 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of intellectual property rights held by Altair Engineering Inc. or its affiliates. Software, other products, and materials of Altair Engineering Inc. or its affiliates are protected under laws of the United States and laws of other jurisdictions. In addition to intellectual property rights indicated herein, such software, other products, and materials of Altair Engineering Inc. or its affiliates may be further protected by patents, additional copyrights, additional trademarks, trade secrets, and additional other intellectual property rights. For avoidance of doubt, copyright notice does not imply publication. Copyrights in the below are held by Altair Engineering Inc. or its affiliates. Additionally, all non-Altair marks are the property of their respective owners.

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

## Altair Simulation Products

**Altair® AcuSolve®** ©1997-2023

**Altair Activate®** ©1989-2023

**Altair® Battery Designer™** ©2019-2023

**Altair Compose®** ©2007-2023

**Altair® ConnectMe™** ©2014-2023

**Altair® EDEM™** © 2005-2023

**Altair® ElectroFlo™** ©1992-2023

**Altair Embed®** ©1989-2023

**Altair Embed® SE** ©1989-2023

**Altair Embed®/Digital Power Designer** ©2012-2023

**Altair Embed® Viewer** ©1996-2023

**Altair® ESAComp®** ©1992-2023

**Altair® Feko®** ©1999-2023

**Altair® Flow Simulator™** ©2016-2023

**Altair® Flux®** ©1983-2023

**Altair® FluxMotor®** ©2017-2023

**Altair® HyperCrash®** ©2001-2023

**Altair® HyperGraph®** ©1995-2023

**Altair® HyperLife®** ©1990-2023

Altair® HyperMesh® ©1990-2023  
Altair® HyperSpice™ ©2017-2023  
Altair® HyperStudy® ©1999-2023  
Altair® HyperView® ©1999-2023  
Altair® HyperViewPlayer® © 2022-2023  
Altair® HyperWorks® ©1990-2023  
Altair® HyperXtrude® ©1999-2023  
Altair® Inspire™ ©2009-2023  
Altair® Inspire™ Cast ©2011-2023  
Altair® Inspire™ Extrude Metal ©1996-2023  
Altair® Inspire™ Extrude Polymer ©1996-2023  
Altair® Inspire™ Form ©1998-2023  
Altair® Inspire™ Mold ©2009-2023  
Altair® Inspire™ PolyFoam ©2009-2023  
Altair® Inspire™ Print3D ©2021-2023  
Altair® Inspire™ Render©1993-2023  
Altair® Inspire™ Studio ©1993-2023  
Altair® Material Data Center™ ©2019-2023  
Altair® MotionSolve® ©2002-2023  
Altair® MotionView® ©1993-2023  
Altair® Multiscale Designer® ©2011-2023  
Altair® nanoFluidX® ©2013-2023  
Altair® OptiStruct® ©1996-2023  
Altair® PolEx™ ©2003-2023  
Altair® PSIM™ © 2022-2023  
Altair® Pulse™ ©2020-2023  
Altair® Radioss® ©1986-2023  
Altair® romAI™ © 2022-2023  
Altair® S-FRAME® © 1995-2023  
Altair® S-STEEL™ © 1995-2023  
Altair® S-PAD™ © 1995-2023  
Altair® S-CONCRETE™ © 1995-2023  
Altair® S-LINE™ © 1995-2023

**Altair® S-TIMBER™** © 1995-2023

**Altair® S-FOUNDATION™** © 1995-2023

**Altair® S-CALC™** © 1995-2023

**Altair® S-VIEW™** © 1995-2023

**Altair® Structural Office™** © 2022-2023

**Altair® SEAM®** © 1985-2023

**Altair® SimLab®** ©2004-2023

**Altair® SimLab® ST** © 2019-2023

**Altair SimSolid®** ©2015-2023

**Altair® ultraFluidX®** ©2010-2023

**Altair® Virtual Wind Tunnel™** ©2012-2023

**Altair® WinProp™** ©2000-2023

**Altair® WRAP™** ©1998-2023

**Altair® GateVision PRO™** ©2002-2023

**Altair® RTLvision PRO™** ©2002-2023

**Altair® SpiceVision PRO™** ©2002-2023

**Altair® StarVision PRO™** ©2002-2023

**Altair® EEvision™** ©2018-2023

### **Altair Packaged Solution Offerings (PSOs)**

**Altair® Automated Reporting Director™** ©2008-2022

**Altair® e-Motor Director™** ©2019-2023

**Altair® Geomechanics Director™** ©2011-2022

**Altair® Impact Simulation Director™** ©2010-2022

**Altair® Model Mesher Director™** ©2010-2023

**Altair® NVH Director™** ©2010-2023

**Altair® NVH Full Vehicle™** © 2022-2023

**Altair® NVH Standard™** © 2022-2023

**Altair® Squeak and Rattle Director™** ©2012-2023

**Altair® Virtual Gauge Director™** ©2012-2023

**Altair® Weld Certification Director™** ©2014-2023

**Altair® Multi-Disciplinary Optimization Director™** ©2012-2023

### **Altair HPC & Cloud Products**

**Altair® PBS Professional®** ©1994-2023

**Altair® PBS Works™** © 2022-2023

**Altair® Control™** ©2008-2023

**Altair® Access™** ©2008-2023

**Altair® Accelerator™** ©1995-2023

**Altair® Accelerator™ Plus** ©1995-2023

**Altair® FlowTracer™** ©1995-2023

**Altair® Allocator™** ©1995-2023

**Altair® Monitor™** ©1995-2023

**Altair® Hero™** ©1995-2023

**Altair® Software Asset Optimization (SAO)™** ©2007-2023

**Altair Mistral™** ©2022-2023

**Altair® Grid Engine®** ©2001, 2011-2023

**Altair® DesignAI™** ©2022-2023

**Altair Breeze™** ©2022-2023

**Altair® NavOps®** © 2022-2023

**Altair® Unlimited™** © 2022-2023

### **Altair Data Analytics Products**

**Altair Analytics Workbench™** © 2002-2023

**Altair® Knowledge Studio®** © 1994-2023

**Altair® Knowledge Studio® for Apache Spark** © 1994-2023

**Altair® Knowledge Seeker™** © 1994-2023

**Altair® Knowledge Hub™** © 2017-2023

**Altair® Monarch®** © 1996-2023

**Altair® Panopticon™** © 2004-2023

**Altair® SmartWorks™** © 2021-2023

**Altair SLC™** ©2002-2023

**Altair SmartWorks Hub™** ©2002-2023

**Altair® RapidMiner®** © 2001-2023

**Altair One™** ©1994-2023

### **Third Party Software Licenses**

AcuConsole contains material licensed from Intelligent Light ([www.ilight.com](http://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/](http://www.altair.com/customer-support/).

### Altair Training Classes

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

### 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	<a href="mailto:anz-pbssupport@altair.com">anz-pbssupport@altair.com</a>
China	+86 21 6117 1666	<a href="mailto:pbs@altair.com.cn">pbs@altair.com.cn</a>
France	+33 (0)1 4133 0992	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>
Germany	+49 (0)7031 6208 22	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>
India	+91 80 66 29 4500 +1 800 208 9234 (Toll Free)	<a href="mailto:pbs-support@india.altair.com">pbs-support@india.altair.com</a>
Italy	+39 800 905595	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>
Japan	+81 3 6225 5821	<a href="mailto:pbs@altairjp.co.jp">pbs@altairjp.co.jp</a>
Korea	+82 70 4050 9200	<a href="mailto:support@altair.co.kr">support@altair.co.kr</a>

Location	Telephone	E-mail
Malaysia	+91 80 66 29 4500 +1 800 208 9234 (Toll Free)	<a href="mailto:pbs-support@india.altair.com">pbs-support@india.altair.com</a>
North America	+1 248 614 2425	<a href="mailto:pbssupport@altair.com">pbssupport@altair.com</a>
Russia	+49 7031 6208 22	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>
Scandinavia	+46 (0) 46 460 2828	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>
Singapore	+91 80 66 29 4500 +1 800 208 9234 (Toll Free)	<a href="mailto:pbs-support@india.altair.com">pbs-support@india.altair.com</a>
South Africa	+27 21 831 1500	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>
South America	+55 11 3884 0414	<a href="mailto:br_support@altair.com">br_support@altair.com</a>
United Kingdom	+44 (0)1926 468 600	<a href="mailto:pbssupport@europe.altair.com">pbssupport@europe.altair.com</a>

See [www.altair.com](http://www.altair.com) for complete information on Altair, our team and our products.

# Index

## Special Characters

[.cshrc](#) 7  
[.profile](#) 7

## A

[Accelerator Administrator Tutorials](#) 31  
[access to help](#) 5  
[add additional environment directories](#) 53  
[add workstation/offhours vovtaskers](#) 47  
[advanced commands](#) 58  
[advanced policy configuration](#) 40

## B

[bin/bash](#) 7  
[bin/csh](#) 7  
[bin/tcsh](#) 7  
[browser-based setup](#) 37

## C

[compose good environment scripts](#) 54  
[configuration examples](#) 42, 54  
[configure an environment](#) 51  
[configure and manage Monitor-basic](#) 60  
[configure default resources](#) 40  
[configure FairShare](#) 38  
[configure policy.tcl for FairShare](#) 38  
[configure resources](#) 42  
[configure security](#) 44  
[configure taskers](#) 46  
[customize actions needed to enable access to Altair Accelerator products on Windows](#) 11

## D

[define a logical HOME directory name](#) 56

## E

[enable Altair Accelerator for non-interactive shells](#) 8  
[enable CLI access on UNIX](#) 7  
[enable CLI access on Windows](#) 10  
[enable the command prompt to communicate with a running product server](#) 12  
[enable the shell to communicate with a running product server](#) 9  
[enforce job resource rules](#) 40



## **F**

find and view example environment scripts [51](#)

find and view resources.tcl file [42](#)

find the server working directory [32](#)

find the URL [37](#)

## **G**

get detailed information about a job [24](#)

get summary information [15](#)

## **H**

help, Accelerator [5](#)

## **I**

invoke the GUI [27](#)

## **J**

job control [21](#)

jobs, show current information [27](#)

## **L**

locate server configuration directory to find policy.tcl [38](#)

locate the security configuration file: security.tcl [44](#)

logical names (equivalences [56](#)

## **M**

method 1: use windows explorer to set command line environment [10](#)

method 2: using Windows command prompt to set command line environment [10](#)

metrics, scheduler [27](#)

Monitor Basic setup [59](#)

monitoring jobs, taskers and resources [25](#)

## **N**

nc\_gui [27](#)

## **O**

online help [5](#)

overlapping queues [62](#)

## **P**

PDF, access [5](#)

## R

rerun jobs [23](#)  
resource management [59](#)  
resource monitor [43](#)  
resource throttling [61](#)  
restart the Accelerator queue [36](#)  
run basic jobs [13](#)  
run jobs with various options [18](#)

## S

save the new configuration [38](#)  
scheduler metrics [27](#)  
security configuration examples [44](#)  
server configuration parameters [39](#)  
setup using the web page [37](#)  
specify name of logfile: -l <logfile> [20](#)  
start a queue [32](#)  
start a test queue [32](#)  
start a vovtasker from the command line [48](#)  
start Accelerator [35](#)  
start newly defined vovtaskers [47](#)  
start/stop Accelerator [35](#)  
start/stop vovtaskers [48](#)  
stop Accelerator [36](#)  
submit multiple jobs at once [18](#)

## T

test the new FairShare configuration [39](#)  
troubleshooting [30](#)  
troubleshooting the UNIX setup [9](#)

## U

upgrade Accelerator [62](#)  
use a specific queue [34](#)  
use Accelerator help [5](#)  
use environments: -e <env> [18](#)  
use of resources: -r <res1 res2 ...> [19](#)  
use the web browser [29](#)

## V

verify access to Altair Accelerator products [7](#)  
verify context is working [11](#)  
view Accelerator status [35](#)  
VOV\_STDOUT\_SPEC [5](#)

vovbrowser [5](#)  
vovbuild [5](#)  
vovconsole [27](#)  
vovdoc [5](#), [5](#)  
vovid [5](#)  
vovserver [5](#)  
vovtasker configuration [46](#)  
vtk\_flexlm\_monitor procedures [61](#)

## **W**

wait for jobs [19](#)