



# ALTAIR

ONLY FORWARD

Altair FlowTracer 2025.1.2

User Guide

# Contents

<b>Altair FlowTracer User Guide</b> .....	5
FlowTracer Roles.....	7
Introduction to VOV.....	9
Concepts and Terminology.....	10
Client/Server Architecture.....	10
Files, Jobs, Nodes, and Sets.....	12
Inputs and Outputs.....	15
Up-Cones and Down-Cones.....	15
Basic Change Propagation.....	20
Sets.....	20
Node Status.....	23
Run, Rerun, Stop and Kill Jobs.....	26
Run and Rerun.....	26
Rerun from the Browser.....	26
Rerun from the GUI.....	27
Rerun from the Command Line Interface.....	28
Rerun the down-cone of a File (kicking a file).....	29
Rerun Modes.....	29
Priority.....	30
Rerun Failed Jobs.....	32
Job Life Cycle During a Run.....	32
Avoid Expensive Reruns.....	34
Check All Files in a Project.....	35
Job Queue.....	36
Dequeue and Stop Jobs.....	38
Kill Processes using the CLI.....	41
Find, Remove and Clean Files.....	43
Show a List of Jobs.....	43
Find Files and Jobs.....	45
Query the vovserver.....	46
Invalidate Dependent Nodes.....	53
Clean Directories.....	55
Remove Nodes From the Graph.....	56
Remove Files.....	59
Remove Large Files That Are Easy to Recreate.....	60
Graphical User Interface.....	62
The GUI Console.....	62
Set Browser.....	63
Set Viewer.....	66
Graphical Tasker LED Monitor.....	66
System Log.....	67

Customize the GUI.....	67
Icons.....	70
Using the Set Browser and Set Viewer.....	71
Keyboard Shortcuts.....	77
Node Selection.....	79
Weight Driven Placement.....	79
Navigate the Flow Graph.....	82
Monitor Taskers.....	83
Browser Interface.....	85
Project Home Page.....	86
Customize the Web User Interface.....	92
Command Line Interface.....	95
List Files in Directory - Showing Project Status.....	97
List Jobs from Project Having a Given Directory.....	99
Start Run - Run Jobs Depending on Changed Inputs.....	100
Show Information About Current Project.....	102
Show the History of a File or Job Within the Project.....	103
Impact of Changing a File or Node.....	105
Why a Node Has its Current Status.....	106
List Inputs and Outputs of a Node for Navigation.....	107
Clean Out Stuck, Unproductive Nodes.....	108
Run Tcl Scripts that Use Tcl Extensions.....	109
vovps.....	110
Search Log Files.....	111
Create Sets with Selection Rules.....	113
Set Names.....	113
Selection Rules.....	113
Regular Expressions in Selection Rules.....	117
Time Specifications.....	117
Advanced Information.....	119
Tasker: vtk_tasker_set_timeleft.....	119
Troubleshoot Taskers.....	120
Manage Remote Taskers Without SSH/RSH Capabilities.....	121
vovtsd on UNIX.....	123
Troubleshooting.....	124
Troubleshooting the Retrace.....	124
Create a FlowTracer Project.....	126
Project Creation Overview.....	126
Each Project Needs a Server Host.....	127
Project Creation Command.....	127
Start the vovserver.....	130
Connect to a Project via the Command Line.....	131
Troubleshooting: Cannot Enable Project.....	132
Shutting Down a Project.....	132
Destroy a Project.....	134
Sanity Check for vovserver.....	134
Resource Management.....	137

Rename Nodes.....	140
Miscellaneous.....	142
Enable CLI Access on UNIX.....	142
Enable CLI Access on Windows.....	145
FlowTracer For UNIX Users.....	148
Reports.....	156
<b>Legal Notices</b> .....	158
Intellectual Property Rights Notice.....	159
Technical Support.....	165
<b>Index</b> .....	166

FlowTracer is a complete system to create, manage, and execute design flows.

This chapter covers the following:

- [FlowTracer Roles](#) (p. 7)
- [Introduction to VOV](#) (p. 9)
- [Concepts and Terminology](#) (p. 10)
- [Run, Rerun, Stop and Kill Jobs](#) (p. 26)
- [Find, Remove and Clean Files](#) (p. 43)
- [Graphical User Interface](#) (p. 62)
- [Browser Interface](#) (p. 85)
- [Command Line Interface](#) (p. 95)
- [Create Sets with Selection Rules](#) (p. 113)
- [Advanced Information](#) (p. 119)
- [Troubleshooting](#) (p. 124)
- [Create a FlowTracer Project](#) (p. 126)
- [Resource Management](#) (p. 137)
- [Rename Nodes](#) (p. 140)
- [Miscellaneous](#) (p. 142)
- [Reports](#) (p. 156)

A *design flow* is a collection of interdependent jobs used to carry out design tasks. Design flows are an important corporate asset that need to be captured and protected. Design flows can be found in projects involving chip design, chip testing, library characterization, software development, and regression testing, among others. Basically, they can be found anywhere where there are two or more jobs that need to be executed in a well defined order to accomplish a design task. Such a task might require compilation then the formation of a link, placement then routing, or the execution of a test followed by report generation. Large flows may contain hundreds of sequential steps, while others may consist of tens of thousands of interdependent jobs.

FlowTracer captures and updates the design flow by interacting with the tools while they are executing. This unique technique to manage the dependencies between files and tools is called *runtime tracing*.

FlowTracer provides the following services:

## **Design management services**

- Automatic management of [design file consistency](#)
- Coordination of [concurrent activities](#)
- Automatic warnings when a tool tries to read invalid data
- Documentation of the design flow
- Reuse of design methodologies

- Discovery of parallelisms
- Automatic execution of tools

### **Intelligent change propagation**

- Non-significant changes are detected and stopped from propagating
- Patented techniques (see [Runtime Change Propagation Control](#))

### **Network computing services**

- Interface with FlexNet Publisher and other licensing systems
- Interface to local compute farm
- Capability to directly manage multiple [Taskers](#)
- Priority scheduling with FairShare
- Load sharing
- Load balancing
- [Queuing](#)
- Near-zero latency in job dispatching
- Hardware and software resource management

## **Value Proposition**

The value proposition for FlowTracer boils down to:

- A faster, safer design cycle
- Higher utilization of available computing resources
- Higher quality flows

# FlowTracer Roles

When using FlowTracer in large deployments, three main roles emerge: Administrator, Developer and User. In smaller organizations, a given person may have multiple roles, but the duties of each role still exist.

The FlowTracer Administrator will install and configure the product and help other users to access it. FlowTracer Administrator enables effective use of FlowTracer:

- Installing FlowTracer
- Installing needed licensing
- Configuring FlowTracer for the whole company
- Managing local files
- Establishing security rules
- Defining user roles
- Helping users set up command line environment

A FlowTracer Developer will create and maintain the job definitions and dependency rules for different applications that form the basis for a flow that is registered with FlowTracer to manage.

FlowTracer Developers use the features that define and manage flows:

- Makefiles
- Tcl Scripts
- Flow Description Language (FDL)
- Enabling runtime tracing
  - Encapsulation
  - Instrumentation
  - Interception
- vtk\_\* Application Programming Interfaces (APIs)
- Command Line Interface (CLI) programs that manage a flow

A FlowTracer User is the recipient of flows pre-developed by a developer. The user relies on the standardization and accuracy provided by FlowTracer to increase their productivity. Typically, they will:

- Execute flows that have been prepared by somebody else.
- Change input files and ask FlowTracer to run the dependent jobs that generate new outputs (this is called "retracing").
- When using the FlowTracer console, the user watches the job nodes changing from purple to yellow to green. When the whole set of nodes is green, the full set of jobs has run and the results can be checked.
- If a job node turns red (FAILED), the user needs to investigate the failure: is it a real failure caused by a problem in the input data, or is it caused by external causes such as disk full, missing licenses, bad machines, dependency violations, or whatever else?
- If the jobs appear to be stuck on some status (think of "color"), the user investigate the cause.

This manual is for topics related to the usage of FlowTracer with pre-existing flows.



## Introduction to VOV

FlowTracer is derived from the system "VOV" originally developed at UC-Berkeley. The word "VOV" is not an acronym, but can be thought of as a contraction of the word "evolve".

Because of this heritage, a lot of the utilities in FlowTracer have a name that begins with either "vov" or with "v". For example, the main program used to manage a flow is called "vovserver", the main clients to connect to the server are called "vovsh" and "vovtasker", and the tool to convert a Makefile into a flow is called "vovmake".

# Concepts and Terminology

## Client/Server Architecture

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

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

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

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


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



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

A typical FlowTracer configuration looks like the following:

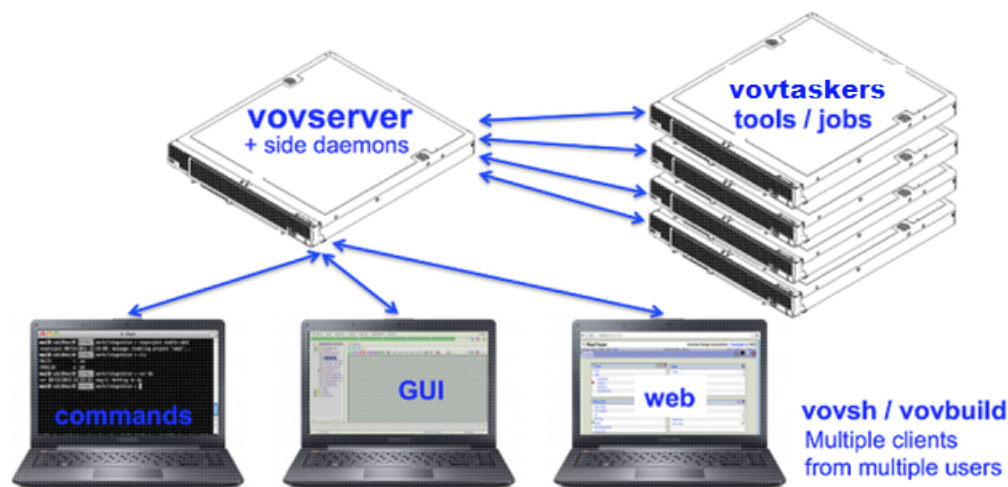


Figure 2:

The elements are:

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

## VOV Projects

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

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

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

## Relationship between FlowTracer, Accelerator, and Monitor

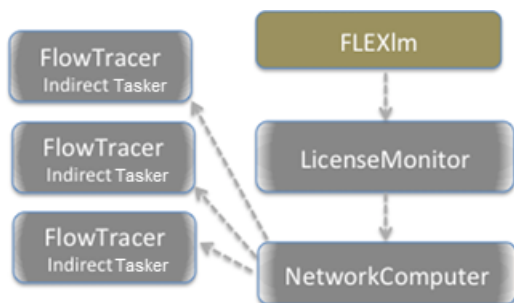


Figure 3: The Interaction of FlowTracer with Accelerator

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

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

## Files, Jobs, Nodes, and Sets

All FlowTracer services are based on the management of the dependencies between files and jobs.

These dependencies are represented in a dependency graph, called *the runtime trace*, *the design flow* or simply *the flow*. The nodes of the graph represent either files or jobs.

In the GUI display of the flow graph, the job nodes are represented by rectangles, while the file nodes are represented by either ovals or hexagons. The graphical representation of the direction of the flow

dependency is either from top to bottom (in the vertical view) or from left to right (in the horizontal view).

The flow graph looks like this:

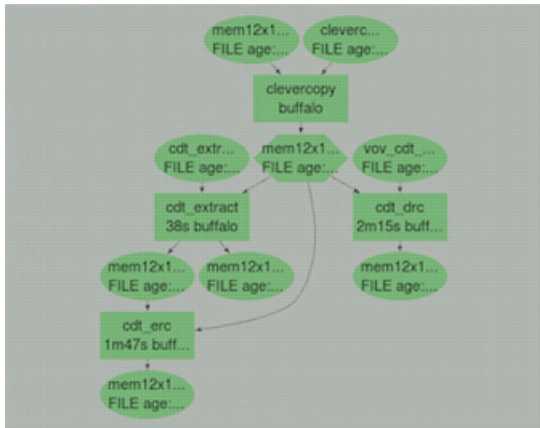


Figure 4: A Small Flow

## Files Represent Design Data

When a project is a chip design flow, the design data for a project consists of the collection of all "files" in the design flow. A "file" in FlowTracer has a very broad definition: it is *any timestamped named entity in a persistent database*.

The main attributes of a FlowTracer file are the:

- [Persistent Database](#) to which the entity belongs
- Names of the entities in the database
- Timestamp to indicate when the entity was last modified.

Normally, a file represents a regular UNIX or Windows file:

- The persistency is provided by the file system itself
- The database attribute is set to FILE
- The name of the file is a canonical name, which is related to the full path for the file. The [Canonical and Logical File Names](#) is automatically computed by FlowTracer.
- The timestamp is the modification time of the file, as reported by the file system.

A FlowTracer file may also represent entities other than files, such as a symbolic links, or URL in the internet.

## Jobs Represent Actions

A job represents a particular invocation of a program that operates on files. Each job is characterized by three attributes that make it unique:

- The [environment](#) in which it is executed
- The working directory
- The command and its arguments

and by a large number of secondary attributes, including the:

- User, or owner, of the job
- Operating system group of the job
- FairShare group the job belongs to
- [Resource list](#)
- Legal exit codes
- Jobname
- Jobclass

and by several computed attributes that cannot be modified, including the:

- Host name where the job has been executed
- Execution start time
- Execution end time
- Expected duration
- Exit status

## Nodes

Both files and jobs are represented by nodes of the flow graph. They share a number of important attributes. Each node has a [VovId](#), which is a nine-digit string such as "000012345".

Each node also has a [status](#), which indicates whether the node is up-to-date with respect to all its dependencies.

Each node also has a *NodeTimestamp*. This is different from the timestamp of a file and from the start or end date of a job. The NodeTimestamp represents the last time in which the node has changed status.

## Sets of Nodes

A typical flow has thousands of nodes. To ease the management of this complexity, FlowTracer supports the notion of "sets of nodes." Each set has the following attributes:

- [Name](#)
- [Selection Rules](#)
- Time of creation
- Owner (not used)

The name of a set can be any unique string. The selection rule, if given, is a rule that is used to determine the membership of nodes in the set. The time of creation and the owner are managed automatically by FlowTracer.

Sets are used extensively in FlowTracer. Some sets are internal to FlowTracer; they can be recognized by their name, which begins with "System:". For example, the set "System:nodes" is the set of all nodes in the flow.

With the browser interface, you can either browse the set hierarchy or get a complete alphabetical list of all sets. You can also browse the set hierarchy in the FlowTracer GUI console.

## Inputs and Outputs

The dependencies are represented by connectors between files and jobs.

The connectors go from files to jobs for inputs and from jobs to files for outputs. For a job, the notion of its inputs and outputs is obvious. We extend the same notion to files: the "outputs of a file" are all the jobs that use the file as input; the "inputs of a file" are all the jobs that contribute to creating the file. Normally, each file has either zero or one input.

A *primary input* is a node that has no inputs. Similarly a *primary output* is a node that has no outputs. In the attached diagram, the files `aa` and `cp` are primary inputs and the files `cc` and `dd` are primary outputs.

Connectors have no attributes.

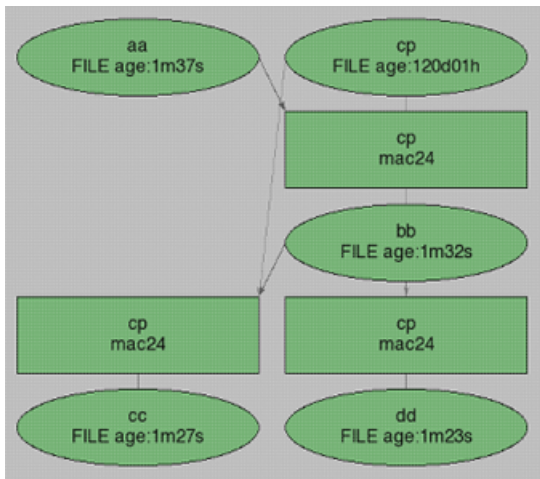


Figure 5: A Small Flow Shows Dependencies Between Files and Jobs

## Up-Cones and Down-Cones

The set of all transitive inputs of a node, that is the set consisting of all inputs of a node and all their inputs and so on, is called the up-cone of a node. The node itself is the apex of, and belongs to, its own up-cone.

Similarly, the down-cone of a node is the set of all the transitive outputs of the node, including the node itself.



**Note:** The commands used in the following examples include the UNIX command `cp` (copy) and `vw` vov wrapper.

## Examples of Up-cone and Down-cone

Here is the graph of a simple flow which contains 3 jobs:

1. vw cp aa bb
2. vw cp bb cc
3. vw cp bb dd

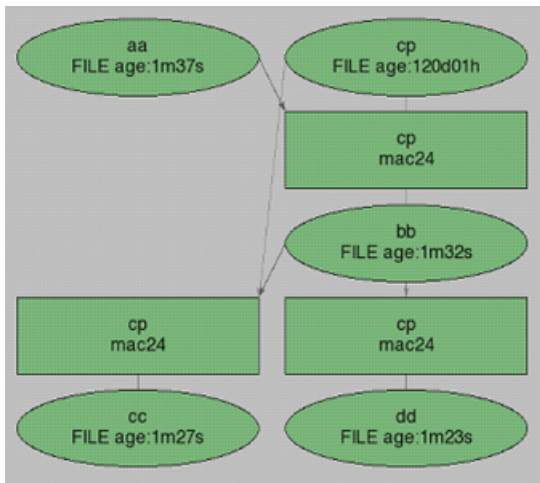


Figure 6:

The down-cone of **node** aa will be:

1. job vw cp aa bb
2. file bb
3. job vw cp bb cc and vw cp bb dd
4. file cc and dd

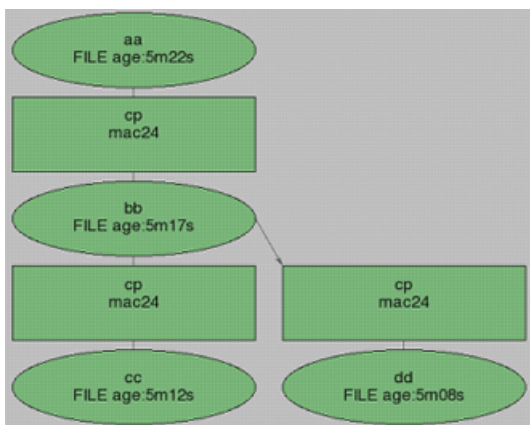


Figure 7:

Similarly, the up-cone of job vw cp bb dd will be:

1. file aa and /bin/cp



2. job vw cp aa bb
3. file bb

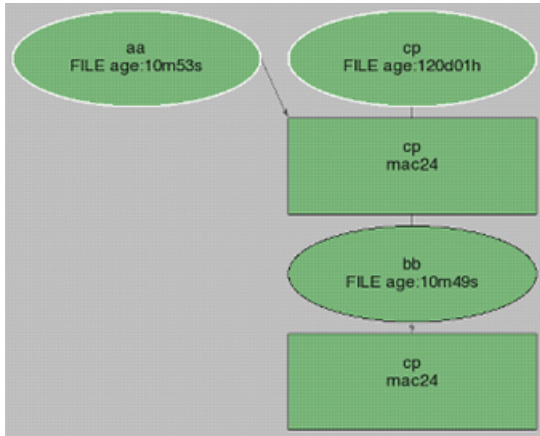


Figure 8:

A typical up-cone or down-cone could be more complex. Here is an example of up-cone of a file:

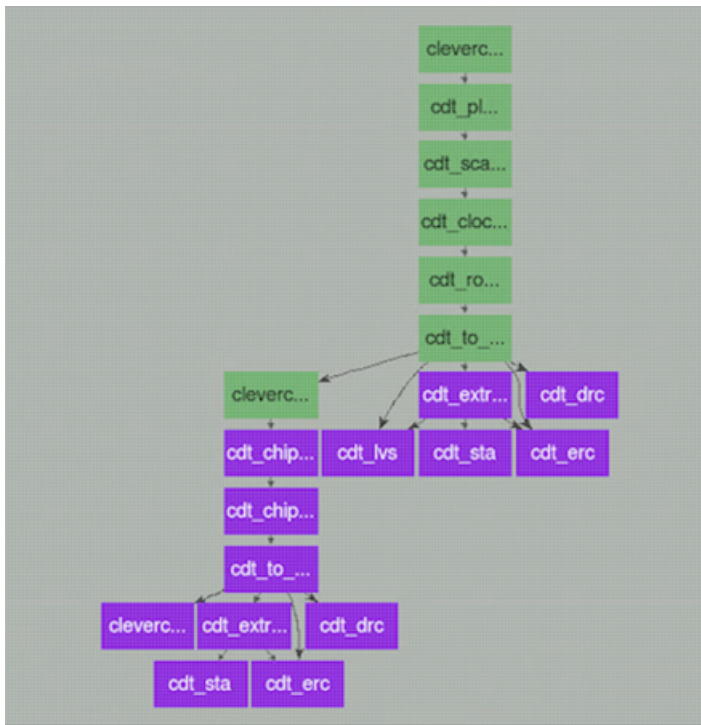


Figure 9:

Here is another example of down-cone of a job:

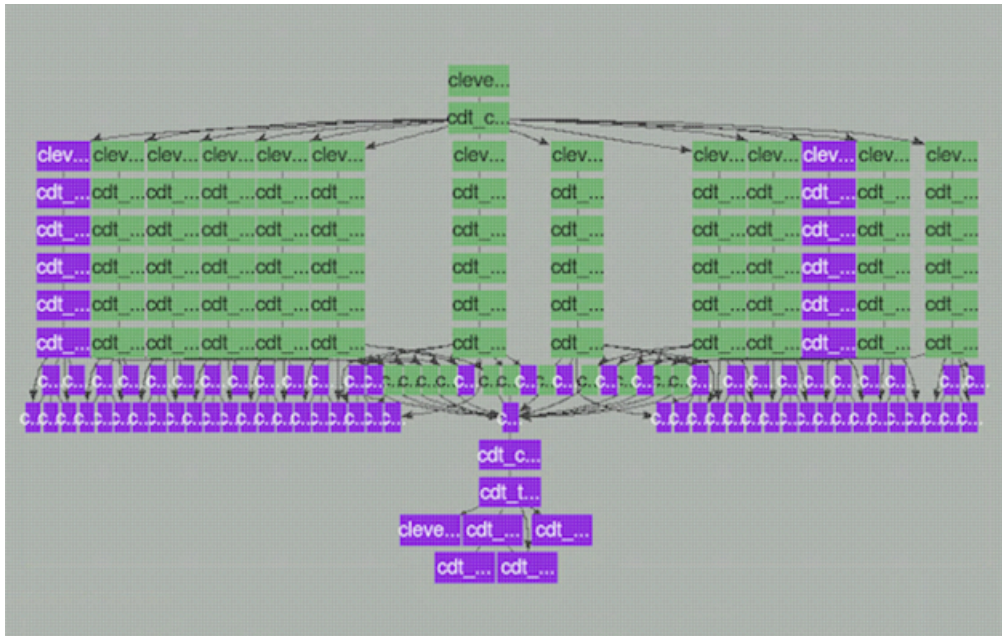


Figure 10:

## Compute Cones with vovcone

Either the graphical user interface or the utility `vovcone` can be used to compute the up-cone or down-cone of a set of nodes in the graph.

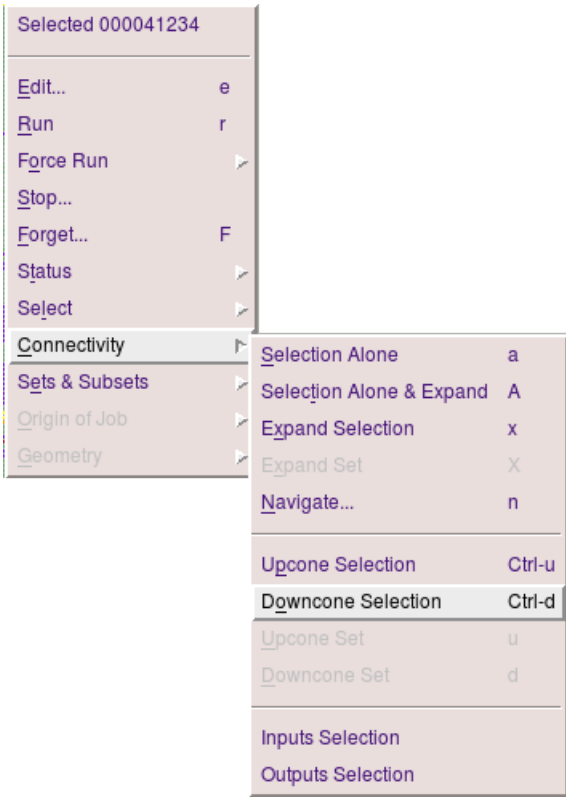


Figure 11:

**vovcone**

vovcone: Usage Message

DESCRIPTION:

Apply a set of upcone and downcone operations starting from one or more jobs or files. For every up/down cone operation, a list of filters (selection rules) can be applied. The resulting set can be saved if a result setname is provided. The content of the result can be formatted.

OPTIONS:

-children

-- Same as -outputs.

-down

-- Start a downcone operation.

-file FILE

-- Specify the starting file (may be repeated).

-filter

-- Define a filter for the current cone operation.

-format

-- Specify output format (default is no output)

-h

-- Help usage message.

-id ID

-- Specify the starting id (may be repeated)

-inputs

-- Start a "inputs" operation.

-job JOBID

-- Specify the starting job (may be repeated).

-O

-- Same as -format

-outputs

-- Start a "outputs" operation.

-parents

-- Same as -inputs.

-result

-- Define name of result set.

-sel

-setname

-- Specify the starting setname (may NOT be repeated).

-up

-- Start an upcone operation.

```
-v          -- Increase verbosity.
```

NOTE:

```
-inputs and -outputs move just one step up and down the  
dependency tree, while -upcone and -downcone move all the  
way up and down the dependency tree.  
-filter is reset for each new operation.  
-format and -result are applied only to the final result.
```

EXAMPLES:

```
% vovcone -file aa -up -format "@ID@ @LEVEL@ @NAME" -filter "isfile"  
% vovcone -job 00012345 -up -format "@ID@ @LEVEL@ @NAME" -filter "isfile"  
% vovcone -file aa -up -filter "status==VALID" -filter isfile  
                  -down -filter "isjob status==RETRACING"  
                  -format "@ID@ @STATUS@ "
```


## Basic Change Propagation

In this section, assume that you have built a flow of any type using any technique. In the flow, there are some files that are primary inputs.

When a primary input file is modified, all the dependent nodes are marked as **INVALID**, to indicate that the design is now inconsistent.

To regain consistency, the change must be propagated to all INVALID nodes. This means that the file's dependent jobs must be run, which in turn create changes in their outputs, which leads to their dependent jobs being run, and so on, until the change propagates to a final job and final outputs.

This process of change propagation is called **retracing** the dependency graph. It is performed by FlowTracer upon user request. FlowTracer processes the request by dispatching the jobs to the taskers in the correct order.

 **Note:** From now on, retracing will be referred to as the easier to remember "run".

During a run, parallelism is exploited, and network resources are used efficiently in order to minimize the total turnaround time.

## Sets

### Sets of Nodes

The nodes in the dependency graph can be grouped into named sets. The names of the sets are unique. It is not allowed to have two sets with the same name.

Sets have many uses in VOV:

- They are used by the server to maintain the internal organization of the graph
- They are used to focus the attention on the problem of interest
- They are used to generate reports
- They are used to define retrace targets.

## Set Status

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

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

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


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

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

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

## vovset

Perform basic operations on sets of nodes.

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

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

```
vovset: Usage Message
DESCRIPTION: 1
    Perform basic operations on sets of nodes.

USAGE:
% vovset attach      <setName> <nodeId> ...
% vovset clear       <setNameOrId>
% vovset count       <setNameOrId> [<status>|<status>+<status>|ALL] ...
% vovset create      [-smart|-smartc] [-universe <setName>] <NewName>
                    <Rule>
% vovset detach      <setName> <nodeId> ...
```

```
% vovset exists      <setName>
% vovset geo         <setNameOrId> [-v] [-h] [-f <file>] [-forget] [-raw]
% vovset info        <setNameOrId> ...
% vovset list        [-rx <filter>] [-all|-names|-ids]
% vovset memberof    <Id> ...
% vovset forget      <setNameOrId> ...
% vovset recompute   <setNameOrId> ...
% vovset rename      <setNameOrId> <newName>
% vovset resources   <setNameOrId> <newResourceList>
% vovset retrace     <setNameOrId> ...
% vovset show        [-rx <filter>] [-sort] [-O <format>]
                    <setNameOrIdOrRule> ...
% vovset stats       <setNameOrId> ...
% vovset tat         <setName> [-O raw]
% vovset union        [-clear|-keep|-append] <largeSetResult> <smallset1> ...
% vovset upcone      <srcSetName> [dstSetName]
% vovset downcone    <srcSetName> [dstSetName]
% vovset zippable     <setNameOrId> <0|1>
```

#### ACTIONS:

```
attach      -- Attach elements (nodes or subsets) to a set.
              Create the set if necessary.
clear       -- Detach all nodes from set.
              New set is created (empty) if necessary.
count       -- Count nodes of a given status in the set.
create      -- Create a new set with given name and rule.
detach      -- Detach elements from a set.
exists      -- Returns 0 or 1 depending on existence
              of set specified by name.
geo         -- Get or set geometry information for elements
              of the set.
info        -- Get basic info about a set.
list        -- List all sets.
forget      -- Forget given sets.
memberof    -- Show IDs and names of sets to which each ID belong
invalidate  -- Invalidate the given sets.
recompute   -- Recompute the given sets based on their
              selection rule.
rename      -- Rename a set.
resources   -- Change resources of jobs in set.
retrace     -- Retrace given sets with
              NORMAL priority in SAFE mode.
              Use vsr to change mode and/or priority.
show        -- Show elements in a set given by name, by id,
              or by selection rule.
stats       -- Get statistics about a set.
tat         -- Get Turn-Around-Time and other stats
timereport  -- Undocumented action -- do not use.
union       -- Create a set from the union of zero or
              more sets to a set.
              If option -clear is used, the result set is
              cleared before the union operation (default)
              If option -keep is used, the result set is kept.

upcone      -- Get IDs of the upcone of the set members.  When
              destSetName is given, upcone elements become members.
downcone    -- Like upcone, but gets the downcone of set members.
zippable    -- Set or reset the zippable flag in all
              files in the set.
```

Action verbs are case insensitive and can be abbreviated to the shortest unique sequence, except "forget" "create" and "recompute" which must be spelled out completely.

#### OPTIONS:

```
-rx <regexp>    -- Regular expression to filter output.
-rule <selrule> -- Apply a selection rule in 'show'.
-O <format>     -- Format of output for 'show'.
-sort          -- Sort output for 'show'.
```

#### EXAMPLES:

```
% vovset attach "MySet" 2345 2468
% vovset clear "MySet"
% vovset count "MySet"
% vovset count "MySet" VALID
% vovset count "MySet" VALID RUNNING
% vovset count "MySet" VALID+FAILED SCHEDULED
% vovset count "MySet" ALL VALID RUNNING
% vovset create "MyEmptySet" ""
% vovset create "ToDo" "isjob status==INVALID"
% vovset create -smart "AllFailed" "isjob status==FAILED"
% vovset forget some_set
% vovset geo "MySet"
% vovset geo "MySet" -raw
% vovset geo "MySet" -f file.geo
% vovset list
% vovset list -rx "UPCONE"
% vovset retrace "ToDo"
% vovset show -O '@ID@ @DURATIONPP@' 'System:jobs'
% vovset show 'isjob status==FAILED'
% vovset tat System:jobs
% vovset tat -O raw System:jobs
% vovset union OurSet YourSet MySet
% vovset union -clear OurSet YourSet MySet
% vovset union -keep OurSet YourSet MySet
% vovset zippable "Some_set" 1
```

## Node Status

Status	Color	Description
VALID	Green	<p>If the node is a job, it has been completed successfully.</p> <p>If the node is a file, it is up-to-date with respect to all files on which it depends.</p>
INVALID	Purple	<p>If the node is a job, it has not been run successfully yet or it needs to be run again, because one of its inputs has been modified since the last time the job was executed. If the node is a file, it is normally the output of a job that is not VALID.</p>

Status	Color	Description
SCHEDULED	Cyan	The job is scheduled to be run. It may be already queued or it will go in the queue as soon as all its inputs are ready.
FAILED	Red	The job ran and failed.
RUNNING	Yellow	<p>If the node is a job, it is currently being executed. All the outputs of such a job are also RUNNING.</p> <p>Normally, for each RUNNING job there is also a RETRACING job, in which case, the RETRACING job is the old job, the RUNNING job is the new one. VOV keeps both jobs to efficiently compare the differences in the I/O dependencies.</p>
RETRACING	Orange	If the node is a job, it is currently being retraced, that is it has been dispatched to one of the taskers. All the outputs of such a job are either RETRACING or RUNNING.
TRANSFER	Cream	The job has been sent to another farm and is still queued in that farm.
SUSPENDED	Pink	The job was running (or retracing) and one of the processes belonging to the job is currently suspended.
MISSING	Brown	<p>This status is used for files that cannot be found by the server. With regular files, this status occurs for one of these reasons:</p> <ul style="list-style-type: none"><li>• Files are deleted, renamed, or moved</li><li>• File systems are unmounted</li><li>• The server does not have permissions to read the files</li></ul>
SLEEPING	Black	<p>The sleeping parts of the graph are ignored by the server. Sleeping nodes are used typically for documentation purposes.</p> <p>Sleeping nodes appear automatically if your flow definition contains output conflicts. The sleeping nodes remain in the graph to allow you to debug the conflict.</p>



Status	Color	Description
DELETED	Dark Green	Files that have been deleted deliberately by a job. The tool <code>compress</code> , for example, deletes its input.
UNKNOWN	Chocolate	The server has not yet collected enough information for the node. This status occurs more often when database proxies are used.
EMPTY	White	This status applies only to sets of nodes which are empty, i.e. which contain no nodes.
WITHDRAWN	Gray	A job has been withdrawn after dispatching, for example by the preemption daemon. This is a rare status to observe.

## Run, Rerun, Stop and Kill Jobs

The act of requesting FlowTracer to bring a particular target up-to-date is sometimes called a *Retrace Request* or more simply a *Retrace*.

### Run and Rerun

During a retrace FlowTracer will analyze the current state of dependencies to create an internal model called the runtime trace, which supports the action of scheduling and dispatching of programs to bring all files up to date. The use of the "re" prefix is used even if it is the first time it is requested for a project. This terminology is a technical way of saying the more casual operations "Run" or "Rerun". From now on, a retrace will be referred to with the "Run" notation.

There can be any number of concurrent run requests. FlowTracer makes sure that, although there are multiple run requests, each job is executed only once and in the correct order.

The same icon is used in all interfaces for the Run button. It is the universal Play icon.

You can run a job with a target that is any combination of:

- a single file
- a single job
- a set of nodes
- the jobs in a directory
- or the whole project

When you request a run:

1. FlowTracer checks whether the target is already up-to-date (e.g. it is VALID), in which case nothing happens
2. If the target is not up-to-date, FlowTracer finds all the jobs that need to be executed to bring the target up-to-date
3. Starts dispatching them to the taskers.

You can assign a [priority](#) for a job when it is registered with FlowTracer. You can reset the priority from the GUI using the **Retrace Priority Flags** dialog which is described below.

The behavior of a run request is further controlled by a number of [retrace modes](#), which affect execution and scheduling of jobs in the presence of [barriers](#) and failed nodes.

### Rerun from the Browser

When visiting a page for a set of nodes, you can start a run using the Run bar. The bar allows you to select priority and flags.

To access the Run bar, from the Project Home page, click the **Jobs** link.



Figure 12: The Node Set Page has a Command Bar with a Button to Request a Run

If you are visiting the page for an INVALID job, then you can start a run by clicking on the **Run** icon. This always starts a normal run at normal priority.

<input type="checkbox"/>	Retrace Id	Mode	Status	Priority	Additional Resources	Jobs	Estimated Duration	Requested by...	...from display	Description	Actions
<input type="checkbox"/>	003929609	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1	2h38m	denby	denby1:10.0	tmp:retrace:set:System:nodes	
<input type="checkbox"/>	003942516	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1316	5h45m	denby	denby1:10.0	tmp:retrace:set:System:nodes	
<input type="checkbox"/>	003942579	SAFE	IN PROGRESS	normal.undef		7	19m05s	denby	--NO DISPLAY--	tmp:retrace:set:src:documentation:html:validate:valid_pages	
<input type="checkbox"/>	003942821	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1316	5h25m	denby	denby1:10.0	tmp:retrace:set:System:nodes	
<input type="checkbox"/>	003942877	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1316	5h25m	denby	denby1:10.0	tmp:retrace:set:System:nodes	
<input type="checkbox"/>	003944089	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1316	5h35m	denby	denby1:10.0	tmp:retrace:set:System:nodes	
<input type="checkbox"/>	003944183	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1316	5h35m	denby	denby1:10.0	tmp:retrace:set:System:nodes	
<input type="checkbox"/>	003945309	SAFE/FORCE/CROSSBARRIERS	IN PROGRESS	normal.undef		1316	5h35m	denby	denby1:10.0	tmp:retrace:set:System:nodes	

Figure 13:

You can get a list of all active run requests by browsing retraces.

## Rerun from the GUI

From the **Set Viewer** window in VovConsole, click the **run** button to request a run. Selected nodes are run, and if none are selected, the entire flow graph is run.

You can also use the keyboard accelerators 'r' and 'R' to request a run. These operations apply to the selected nodes. The lower case 'r' accelerator runs at normal priority, while upper case 'R' requests a high-priority run.

In the GUI, set the priority and possibly the run flags by clicking **Trace > Priority & Flags** to start the dialog, shown below.

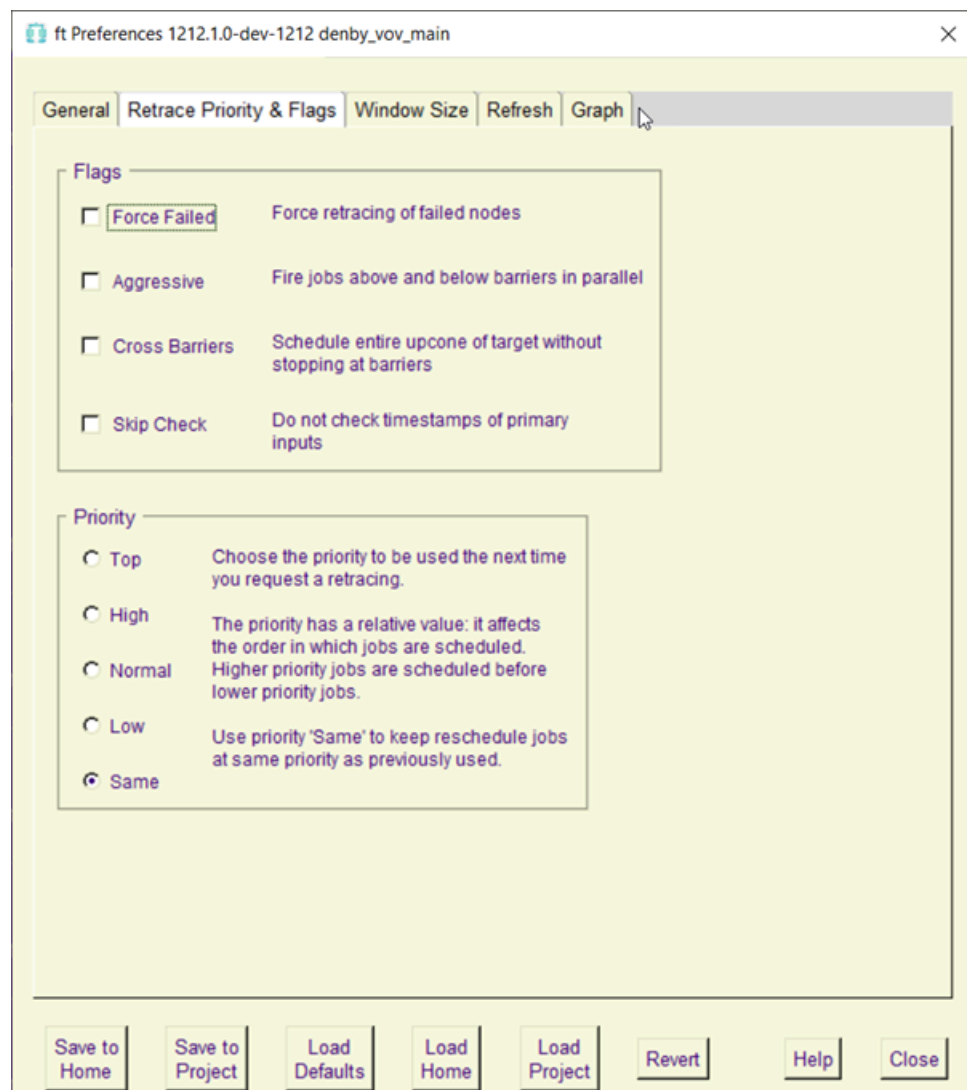


Figure 14:

You can get a list of all active run requests by clicking **Trace > Retrace List**.

## Rerun from the Command Line Interface

From the CLI, you can request FlowTracer to run with the command `vsr`.

You can request a run to a given file with:

```
% vsr fileName_or_VovId
```

You can run all jobs associated with a directory by using:

```
% vsr directoryName
```

You can target an entire set to be run with:

```
% vsr -set setName
```

As a special case, you can run everything in the whole design. Use the private set `System:nodes` with:

```
% vsr -all
```

Set the [priority](#) with the option -p:

```
% vsr -p high directoryName
```

## Rerun the down-cone of a File (kicking a file)

If you change a file and would like to run all jobs that are affected by the change, use:

```
% vsr -kick name_or_VovId_of_changed_file
```

You can get a list of all active runs with the command:

```
% vovshow -retraces
ID      DESCRIPTION                                USER      SIZE MESSAGE
014435890 tmp:retrace:set:tmp:UserSelect john        2
014435949 tmp:retrace:set:System:nodes andrea     1321
014435969 tmp:retrace:set:System:nodes andrea     1322
```

## Rerun Modes

There are several "modes" of running (retracing), having to do with the way jobs are scheduled and the way jobs are executed. The default mode of execution is "SAFE".

Mode	Description
<b>FAST</b>	This flag affects the execution of the jobs. If the execution is FAST, runtime tracing is disabled for the job, meaning that the dependencies are not recomputed.
<b>AGGRESSIVE</b>	<p>This flag affects the firing of the jobs in the presence of <a href="#">barriers</a>. Normally, a job is ready to fire only if its entire up-cone is VALID. If the "AGGRESSIVE" flag is used, a job will fire if all its immediate inputs are VALID. The situation where the immediate inputs are VALID but the up-cone is not VALID normally arises when you have barriers.</p> <p>In AGGRESSIVE mode, jobs are fired in parallel above and below the barrier. If the barrier stops the change propagation, all is OK. Otherwise, the jobs below the barrier will have to be re-executed.</p>

Mode	Description
	The default mode is not AGGRESSIVE, meaning that all jobs above the barrier are executed before the jobs below the barrier.
<b>CROSSBARRIERS</b>	This flag affects the running of jobs in the presence of barriers. Normally, when you run a target, only the jobs below the barriers are run. With the "CROSSBARRIERS" flag, all jobs in the up-cone of the target will be run.
<b>FORCEFAILED</b>	This flag affects the running of FAILED jobs. Normally, FAILED jobs remain FAILED, until the inputs change. With the "FORCEFAILED" flag, the failed nodes in the up-cone of the target are also run.
<b>SKIPCHECK</b>	<p>This flag affects the checking of the primary inputs in the up-cone of the target. Normally, FlowTracer checks the timestamp of all primary inputs, but this could be expensive especially if you are using a slow filesystem such as ClearCase. If you set this flag, the check is skipped.</p> <p>This flag reduces the reliability of the running. You should use this flag in conjunction with alternative ways to check timestamps, such as <code>vls</code> or <code>vovcheckfiles</code>.</p>
<b>SAFE</b>	This is the name for the default mode, the one in which all flags mentioned above are OFF.

## Priority

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

Two types of priorities are supported:

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

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

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

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

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

Symbolic Name	Numerical Level
<b>Top</b>	15
<b>High</b>	8
<b>Normal</b>	4
<b>Low</b>	1

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

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

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

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

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

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

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

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

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

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

## Priorities Relative to Previous Run

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

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


For example:

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

## Execution Priority

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

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

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

## Rerun Failed Jobs

A job is labeled as FAILED if something has gone wrong during the execution of the job.

Chances are that running the job again, without first making the changes necessary to remove the cause of the problem, will result in the same failure. For this reason, FAILED jobs are not normally run.

To run a FAILED job, it must first become **INVALID**. This happens when one of the following occurs:

- One of the dependent inputs of the job changes.
- You explicitly invalidate the job.
- You request a run of the job with the **Rerun Modes** "FORCE".

## Job Life Cycle During a Run

The following diagram shows the normal life cycle for an INVALID job.



Status	Event	What keeps job in this status?
INVALID		If a job remains INVALID even after a run request, it is being blocked by a missing input (run <code>vovforget -blocking</code> ).
↓	A request is issued to perform a runtime tracing analysis for the job.	
SCHEDULED		If a job remains SCHEDULED forever, its inputs are not becoming VALID (investigate what happens to the inputs) or the FlowTracer server does not have access to sufficient resources (check the resource maps and the taskers).
↓	All inputs for the job are VALID and all resources for the job are available and the job is dispatched to a vovtasker.	
RETRACING		
↓	The job successfully fires on the vovtasker (this normally takes a few milliseconds).	
RUNNING		If a job remains RUNNING forever, a problem has occurred during execution. Either the vovtasker died, or the machine died, or more simply, the job is waiting for some user input.
↓	The job completes.	
VALID or FAILED		If the job becomes VALID, all is Ok. If the job fails, you may want to investigate the cause of the failure.

## Avoid Expensive Reruns

Sometimes a complete run may take too much time, and you may want to find ways to bypass an expensive execution, at least in the short term, in order to focus on a more urgent issue.

For example, in the simple flow shown below, there are three jobs in sequence. The second job takes "a long time", and the user is currently focused on the third job.

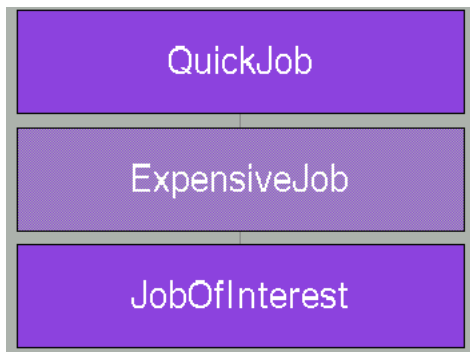


Figure 15: Three Jobs in Sequence Dependency. First and Third Jobs Should be Run, but not the Second

If, for whatever reason, the first job becomes INVALID, you are forced to wait a long time for the second job to finish in order to see work on the third job happen. This may not be acceptable. This section shows some ways to temporarily skip the execution of the long second job. We assume that the job has been executed at least once in the past so that some reasonable instance of dependent inputs are available for the third job.

Only a true run is a guarantee of a correct and repeatable result. The techniques shown in this section are only short-term workarounds to run jobs that are later in a flow sequence for experimental reasons.

### Try Validate

This technique is available only from the GUI. You can select the job to skip and attempt to turn it valid using the context menu, by right-clicking and selecting **Hold > Status > Try Validate**, by clicking **Node > Status > Try Validate** from the top level menu.

After the job turns VALID, you can tell FlowTracer to set its internal state to be consistent by doing a [sanity check](#) command. If the job remains VALID as the internal state is made consistent, then the fake valid state is stable, and the job of interest can be run.

### Force Validate

This technique is available only from the GUI interface. It is a way to fake out the dependency graph analysis by forcing the outputs of a job to appear older than the inputs. It is similar to what the option "-t" does in "make" in order to effectively touch a target that needs to be remade. The nodes that are forcefully validated through this technique are assigned a property called "FORCE\_STATUS\_CHANGE".

The Force Validate technique is dangerous because it may give a false sense of "validity". Check for the property "FORCE\_STATUS\_CHANGE" before considering the flow to be truly up to date. If any nodes exist with this property, then the flow is not actually up to date.

## Using the "skip" Flag

Setting the 'skip' flag is also called turning on 'autoflow'.

The 'skip' flag of a job tells FlowTracer to automatically change the status of the job to VALID as soon as all inputs to the job are also VALID. This happens without any job execution.

Skipped jobs are identified in the GUI by having a slightly different colors than the other jobs.

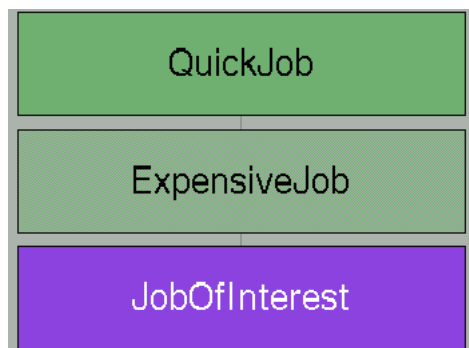


Figure 16: "The Expensive Job" Node has Skip Flag Set to autoflow into VALID State, so the Node is Different Green Color

The skip flag can be used to:

- skip over the execution of some jobs
- waive some tests in a regression.

## Using 'skip' to Skip Job Execution

Sometimes you may want to temporarily skip the execution of a job in order to execute other jobs further down in the flow. If the job is INVALID, FlowTracer will not execute any dependent jobs.

This flag is controlled in many ways:

- Using the command `vovjobskip`
- Using the 'Skip' entry in the pop-up menu in the GUI
- Editing the job with the browser
- Editing the job with the GUI

## Check All Files in a Project

When you request a run, the FlowTracer server checks the timestamp of all primary inputs in the up-cone of the target.

To check all files in the up-cone, instead of only the primary inputs, use the `VOV_VW_CHECK_UPCONE` variable, which tells the wrapper to check all files in the up-cone at run time.

To check the timestamps of all the files in a project, including therefore also all the intermediate files, use `vovcheckfiles`.

```
vovcheckfiles: Usage Message
```

```
DESCRIPTION:
```

```
Check the timestamp of all files in a project.
```

```
This is an expensive operation, so this script  
implements a semaphore to avoid concurrent checks.
```

```
USAGE:
```

```
% vovcheckfiles [OPTIONS]
```

```
OPTIONS:
```

```
-v      -- Verbose  
-h      -- This help
```

```
EXAMPLES:
```

```
% vovcheckfiles -h  
% vovcheckfiles  
% vovcheckfiles -v
```

If the timestamp of any file in the project has changed, for example by being modified by an editor or other program run outside of the flow, the vovserver is notified of the change. This may cause the down-cone of the modified file(s) to be marked INVALID.

The `vovcheckfiles` command tries to prevent more than one instance of itself from running concurrently. See [Troubleshooting](#) for more information.

## Job Queue

For each run request (which calculates the runtime tracing graph), VOV creates a set to contain all the jobs to be executed to satisfy the request. Then VOV takes all the jobs that are ready to execute and places them into the Job Queue in order of priority.

### Job Queue Bucket

Jobs in the job queue are classified into *buckets*. You can think of a "job queue bucket" as a FIFO for jobs that have the same scheduling properties.

The buckets are generated automatically based on the set of jobs in the runtime tracing graph and the number of unique scheduling properties that they share. Each unique group of scheduling properties generates one bucket.

All the jobs in the same bucket share the same values for their scheduling properties: group, user, tool, priority, expected duration, and resources.

During each pass of the job scheduler, the 'top-most' job in each bucket is examined, and if runnable, it is dispatched on a suitable tasker machine.

To show the jobs in the job queue you can use the command `vovjobqueue`. The default is to show the buckets.

```
% vovjobqueue  
#Jobs GROUP      USER      TOOL      WAITING FOR...  
1      1 users     suresh     SERVER_RUN_C linux#1 bison#1 manual#1  
2      2 users     suresh     g++        diskio#1 macosx#1
```

```
3      7 users      suresh  g++      diskio#1 macosx#1
4     14 users      suresh  g++      diskio#1 macosx#1
5      9 users      suresh  g++      diskio#1 macosx#1
6      6 users      suresh  g++      diskio#1 macosx#1
7      6 users      suresh  link     diskio#1 win64#1
8      8 users      suresh  g++      diskio#1 macosx#1
```

or the similar command `vovshow -buckets:`

```
% vovshow -buckets
ID          JOBS PRI FSGROUP      USER  TASKERS  RANK WAITING FOR...
014453389    1   4 /time/users john      0      0 slot(macosex manual)
014457641   10   4 /time/users john      1      0 slot(unix doc)
```

The bucket view of jobs provides an overview of the jobs in a way that is not overwhelming. If you want to see the job details, you can drill down to see the list of jobs in the bucket, and then drill down into any given job.

The WAITING FOR column displays the resource bottlenecks for each bucket. For quantitative resources, the format will be resource#quantity. If the bucket is being held back by the FairShare configuration, the symbolic FS SKIP will be displayed.

## Monitor the Job Queue Buckets

There are two basic displays for the entries in the job queue buckets.

- VOV Monitors' window
- Browser-based display

The VOV Monitors window may be brought up from either the VOV console, or from the command line. It provides tabs to select what type information to view. One tab view is for the Job Queue Buckets.

The following table shows how to display the job queue buckets from various VOV user interfaces:

Interface	Use
<b>command line (CLI)</b>	<code>% vsm -panel buckets</code>
<b>VOV Console (GUI)</b>	<b>Console &gt; Vov Monitor</b> <b>Open Vov Monitors</b> icon on far right of Tasker Monitor row at top
<b>Browser</b>	Job Queue page Job Stats page



SlaveGroups	Slaves	Slave HW	Slave Resources	Whc	Running Jobs	Running Commands	Running Details	Resource	Queued Jobs	Queue Buckets	FairShare
	Pos	Jobs	Group	Age	Tool	Priority	Resources	Exhausted Res.	Missing Res.		
	1	1	1432	/time/users.t	1s	validate	top	hasValidate		slot(hasValidate)	

Figure 17: Vov Monitor Showing the Queue Buckets Tab, which Displays the Job Queue Buckets Panel

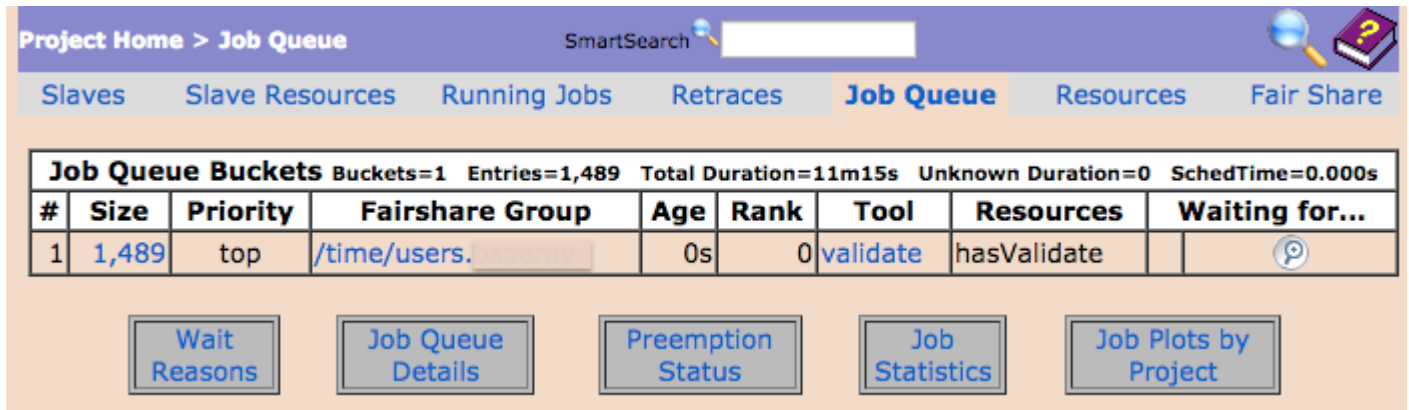


Figure 18: Browser Showing the Job Queue Tab, which Displays the Job Queue Buckets Table

## Dequeue and Stop Jobs

Often we use the verb "stop" in a loose fashion, but in this section it is important to distinguish "dequeue" from "stop". Dequeuing only impacts scheduled jobs and leaves running jobs unaffected, while "stopping" really means killing jobs that are running.

Icon	Action	Description
	Dequeue	Dequeue a scheduled job, ineffective on a running job.
	Stop	Dequeue a scheduled job and stop a running job. The job is stopped by sending in order the signals TERM HUP INT KILL until the job is no longer running.

## Control the Signals to Stop Jobs

Some jobs require specific signals to be delivered in order to be stopped cleanly. If you have such job, you can use the properties STOP\_SIGNAL\_LIST and STOP\_SIGNAL\_DELAY to control the stopping sequence.

Below is a fragment of Flow Description Language that you can use to specify such job:

```
# This works in 2013.09
J vw ./my_job_that_is_hard_to_stop some arguments
vtk_prop_set $make(transId) STOP_SIGNAL_LIST "HUP,USR1"
vtk_prop_set $make(transId) STOP_SIGNAL_DELAY "10s"

# This works in version 2014 and later
J vw ./my_job_that_is_hard_to_stop some arguments
P STOP_SIGNAL_LIST "HUP,USR1"
```

```
P STOP_SIGNAL_DELAY "10s"
```

## Stop Jobs using the Browser Interface

You can stop the run by going to the Retraces page, finding the request you want, and stopping it by clicking on the **Dequeue** icon.

Stopping a run does not stop the jobs that are currently running on the taskers. Instead, it simply stops sending any more jobs to the taskers.

You can find all the `/set/create?name=running&rule=isjob+STATUS==RETRACING` and then selectively stop them by going to each job page, or you can go to the Tasker page and then stop the jobs running on each individual tasker.

From the Projects page, you can perform global operations that affect the whole project:

- stopping all retracing requests with URL `/retraces/stopall`
- stopping all running jobs with URL `/taskers/killjobs`

## Stop Jobs using the GUI

You can stop a run by clicking on **Trace > Stop** and then choosing the run. You can stop jobs and taskers from the Tasker Monitor window, by right-clicking on the tasker. You can stop a job by right-clicking on the job in the **Set Viewer** and select **Stop** from the popup.

## Stop Jobs using the CLI

The main command to stop jobs is `vovstop`.

```
vovstop: Usage Message

DESCRIPTION:
  Stop a list of objects.  The jobs are normally descheduled,
  but if the -f (-force) option is used, then also running jobs
  are stopped.

USAGE:
  % vovstop [options]
  % vovstop [options] objectId ...
  % vovstop [options] setName ...

OPTIONS:
  Most options can be abbreviated to one character.

  -help                -- Print this message.
  -q                   -- Quiet.
  -s                   -- Quiet (same as -q).
  -silent              -- Quiet (same as -q).
  -v                   -- Increase verbosity.
  -f                   -- Stop running jobs also, same as -force
  -force               -- Stop running jobs also, same as -f
  -running             -- Stop running jobs ONLY for the specified
                        jobs or set, do not dequeue jobs.
                        Implies -force.
  -dir <dirname>       -- Stop jobs in a directory.
  -subdirs <dirname>   -- Stop jobs in a directory and all subdirs.
  -downcone            -- Stop also all jobs in the downcone
                        of the specified nodes.
```

```
-retraces          -- Stop all current retrace requests.
-alljobs          -- Stop all jobs.
-project          -- Stop current project.
-signals <SIGLIST> -- Comma separated list of signals to send to the jobs
                    (default sequence is TERM,HUP,INT,KILL )
                    This can be also set with property NC_STOP_SIGNALS
                    or with the environment variables NC_STOP_SIGNALS
                    or VOV_STOP_SIGNALS.
                    Priority: 1. Option -signal
                             2. job property NC_STOP_SIGNALS,
                               STOP_SIGNALS
                             3. env variable VOV_STOP_SIGNALS,
                               NC_STOP_SIGNALS
                             4. default (can be configured as
                               defaultStopSignalCascade in policy.tcl)
                    See also: vovshow -env VOV_STOP_SIGNALS
                             vovshow -env NC_STOP_SIGNALS

-sig <SIGLIST>    -- Same as -signals.
-exclude <PROCLIST> -- List of processes to exclude from receiving the
                    signal.
-include <PROCLIST> -- List of processes to receive the signal.
-delay <s>        -- Minimum delay between signals (in seconds),
                    between 0 and 20s. Default is 3.
                    This can also be set with the property
                    NC_STOP_SIG_DELAY, or with the environment
                    variables NC_STOP_SIG_DELAY or
                    VOV_STOP_SIGNAL_DELAY. If both NC_STOP_SIG_DELAY
                    and VOV_STOP_SIGNAL_DELAY are present in the
                    environment, the value of NC_STOP_SIG_DELAY will
                    be used.
                    Priority: 1. Option -delay
                             2. job property NC_STOP_SIG_DELAY
                             3. env variable VOV_STOP_SIGNAL_DELAY,
                               NC_STOP_SIG_DELAY
                             4. default

-why <REASON>     -- The reason why vovstop is being called. This gets
                    passed to the WHYSTATUS field and property if it
                    causes a status change.

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

The 'objectId' argument can refer to a: job, user, retrace, set,  
bucket, or file

#### EXAMPLES:

```
% vovstop System:jobs -- Stop jobs in specified set.
% vovstop 00023456    -- Stop job by id.
                    The id refers to a job, set, retrace,
                    file, bucket, or user.

% vovstop -dir .      -- Stop jobs in current dir.
% vovstop -subdirs .  -- Stop jobs in current dir
                    and all subdirs.

% vovstop -alljobs
% vovstop -f -alljobs -- Stop and kill all jobs
% vovstop -f -signal HUP,TERM -- Send SIGHUP and SIGTERM to all jobs
% vovstop -f -include sleep -- Stop all "sleep" jobs
% vovstop -f -exclude sleep,ls -- Stop all jobs EXCEPT "sleep" and
                    "ls" jobs
% vovstop -retraces -why "The reason why all retraces were stopped"
```



## Kill Processes using the CLI

In order to kill a job, you must be on the machine on which the job is running, identify the PID of the job and then send the Kill signal to the PID.

In contrast, you can stop a job from any machine, because you are essentially telling the vovserver to tell the vovtasker to kill the job.

You can kill the job using kill directly, but you may find the utility `vovkill` to be quite useful because it kills not only a process but also all of its children.

In order to identify the PID of the process to kill, you may also consider using the utility `vovps` to identify the children and parents of a process.

### vovkill

This utility is used to kill processes.

```
vovkill: Usage Message

DESCRIPTION:
  This utility is used to kill processes,

  It differs from kill because:
  - It kills a process and all of its children;
  - It can send a list of signals to try to kill a process.

USAGE: vovkill [options] <pid> ...
  Kill the given process and all of its children.
  The processes are first sent TERM, then HUP, then INT, then KILL.

OPTIONS:
  -v                -- Increase verbosity.
  -n                -- Just show what you would do but do not do it.
  -rx <rx>          -- Kill processes that match the given rx.
  -exclude <pid>    -- Exclude pid from list of processes to kill.
  -pid <pid>        -- Process to kill.
  -signal <s>       -- Which signal you want to send.
  -gentle           -- Send only HUP.

EXAMPLES:
  % vovkill 22123
  % vovkill -signal INT 22123
  % vovkill -gentle 22123
  % vovkill -rx grep -signal 9 223
```

## vovps

vovps lists all processes, similar to the UNIX command `ps`. It is useful to understand the process hierarchy of Accelerator jobs, and sometimes to find which PIDs to kill in a shell.

The `-c` and `-p` options show the child processes and parent processes respectively of a given pid. The results are a Tcl list of pid, program name, and owner.

```
usage: vovps [-an] [-c pid] [-dgsSJj] [-p pid] [-mP]
  -a:      Print all processes, one per line, like a simple version of 'ps'
  -n:      Only print pids: must come before -c or -p
  -c:      Print the children of process <pid>. The printing is in Tcl
           format, but it is also easy to parse.
  -d:      Debug process info collection
  -g:      Include list of processes with same PGID to children
  -s:      Include list of processes with same SID to children
  -S:      Include state of process R,S,T,Z...
  -J:      Include list of processes with same VOV_JOBID to children
  -j:      Like -a, but show the VOV_JOBID also and LM_PROJECT instead of
           command line
  -p:      Print the parents of process <pid>. Same format as for -c
  -m:      Print more detailed memory usage information (Linux).
  -P:      Use 'ps' program to scan the processes. This behavior can also
           be selected by setting the environment variable VOV_USE_PS.
```

## Examples

Example 1:

```
% vovps -a | grep fire
31155 28981 1h06m 0 0 0.00 0.01 bkring vovfire28981.cs
32381 32260 56m35s 977 219 6.45 70.33 bkring firefox
```

Example 2: (the shell substitutes the current pid for '\$\$'):

```
% vovps -p $$
set parents(28756) {
  28754 "xterm" cadmgr
    1 "init" root
}
```

# Find, Remove and Clean Files

## Show a List of Jobs

The tool `vovshow` is used to show things like jobs, retrace requests, the project itself.

```
vovshow: Usage Message

DESCRIPTION:
  Show various types of objects in a VOV project,
  as well as documentation.

USAGE:
  % vovshow [OPTIONS]

OPTIONS:
  -h                -- This help
  -v                -- Increase verbosity
  -w                -- Wide output

Files:
  -missing          -- Show missing files
  -blocking         -- Show blocking files

Jobs:
  -running          -- Show running jobs
  -scheduledjobs    -- Jobs that are scheduled to be run
  -recentjobs       -- Jobs that have recently completed
  -failed           -- Show failed jobs
  -sleeping         -- Show sleeping jobs
  -O format         -- Specify a format string for jobs

Project:
  -alerts           -- Show current alerts
  -buckets          -- Show job queue buckets (see also -jetstreams)
  -clients [type:rx] -- Show server's clients
  -emptysets        -- Show empty sets
  -fairshare        -- Show basic fairshare information
  -fsgroups         -- Same as -fairshare
  -groups           -- Same as -fairshare
  -innerlooptimers  -- Show info about inner loop timers
  -jetstreams       -- Show info about buckets and jetstreams
  -licenses         -- Show the licenses used by vovserver
  -mmchunks         -- Show info about memory allocation
  -parameters       -- Same as -policy
  -policy           -- Show all policy settings
  -project          -- Show project info
  -queries          -- Show all active queries
  -resources        -- Show resource maps
  -reservations [type] -- Show reservations, type can be tasker,
                        resourcemap, or emulator
  -retraces         -- Show retrace requests
  -serverenv        -- Show environment variables used by vovserver
  -tasks            -- Show tasks using 'vovtaskmgr show'
  -users            -- Show users
```

```
-waitreasons      -- Show wait reasons

License based Resources:
  -byjobusage LICENSE -- Show license resource usage by job,
                        assumes Type 'License'

Documentation: (search strings are case sensitive)
  -api              -- Show usage info about VTK and TCL API
  -envvars ES       -- VOV environment variables containing ES
  -failcodes        -- Show the known failure codes for jobs
  -fields           -- Show known fields
  -fieldshtml       -- Show known fields in formatted html
  -prop PS          -- VOV properties containing PS

OPTIONS FOR Monitor:
  -accounts         -- Show default accounts table
                    which is managed by ftlm_lmproject and
                    ftlm_accounts setlive
  -expirations      -- Show features and expirations
  -features         -- Show features
  -licdaemons       -- Show license daemons

OPTIONS FOR Allocator:
  -la               -- Show LA summary
  -lares <RESOURCE> -- Show detail for specified resource

OPTIONS FOR Accelerator:
  -preemptrules     -- Show all preemption rules
  -preemptmethods   -- Show all known preemption methods

EXAMPLES:
% vovshow -running
% vovshow -w -running
% vovshow -emptysets -blocking
% vovshow -envvars NC
% vovshow -prop PREEMPT
% vovshow -retraces
% vovshow -w -retraces

% vovshow -features -tag SNPS
% vovshow -licdaemons
% vovshow -clients 'nickname:.*d$'; # clients where nickname

% vovshow -api vtk_s;                # List all procs that
% vovshow -api vtk_set_get_elements; # Show usage of

% vovshow -la queued
% vovshow -lares License:abc

SEE ALSO THESE COMMANDS THAT SHOW INFORMATION:
vovjobqueue,      vovset list,      vovfind,
vel,              voveventmon,      vovshowconnection,
vovproject list,
vovtaskermgr show
vsi, vls, vst,
vovdoc
```

In a FlowTracer context, the most useful commands are those that show the missing files:

```
% vovshow -missing
% vovshow -blocking
```

and those showing jobs in various status:

```
% vovshow -running
% vovshow -failed
% vovshow -recentjobs
```

## Find Files and Jobs

### Find Files

The tool `vovfind` searches the flow for the files that match the regular expressions provided as arguments.

Each argument is interpreted as a regular expression `RX` and is used to construct [Selection Rules](#) of the form:

```
isfile name~RX
```

For example:

```
% vovfind stdout
00005962 VALID    ${TOP}/Design/.stdout_musa10142542
00006227 VALID    ${TOP}/Design/.stdout_wolfe7515419
00006289 VALID    ${TOP}/Design/.stdout_wolfe5609764
% vovfind c$ h$
00006229 VALID    ${TOP}/Design/main.c
00006289 VALID    ${TOP}/tech/scmos/maniac
00006213 VALID    ${TOP}/Design/local.h
```

### Find Jobs

There is no simple command to find jobs in the flow. Instead, use the more general search mechanism available by means of the command `vovset`.

You can create a temporary set that will hold all the jobs that match a certain criteria, and then display the set with the desired output format. For example:

```
% vovset create "tmp:test" "isjob age<3600"
vovset: message: Created Set Id=00004444 Name=tmp:test Rule=isjob age<3600
% vovset show tmp:test
00012394 INVALID  vw cdwrite -e -v iso9660/ftnc5.1
00012634 INVALID  vov ./printlabel ftnc.ps ../iso9660/ftnc5.1
00017410 INVALID  vrt cdrecord -eject -v speed=2 dev=0,2,0 i...
00392367 INVALID  vov ./build.tcl index.tcl > build_index.log
00392387 INVALID  vov ./makeindex.tcl
00392424 INVALID  clevercopy work/index.html ../../doc/html/...
% vovset show -O '@LEVEL@ @ID@' tmp:test
32 00012394
32 00012634
36 00017410
```


```
8 00392367
4 00392387
6 00392424
```

If you know in which directory a job is supposed to run, you can change to that directory and use `vst -a` to see all the jobs in there.

```
% vst -a
2 014426579 VALID      BASE      vovbuild -T -l topLevelTasks -f TaskFlow.tcl
```

## Query the vovserver

The `vovselect` command provides a way to retrieve specific data from the vovserver, with filtering done on the server side. This method differs from some of the VTK calls, which get all data and require processing on the client side to get the data of interest.


 **Attention:** In an upcoming major release of Accelerator Products, the `vovselect *` wildcard select feature will be dropped. To prepare for this change, users should update scripts and REST requests to issue `vovselect` requests using a specified list of field names. For example:

```
nc cmd vovselect statusnc,id,command from jobs
```

An easy way to find out what fields are in an object type is by using `vovselect fieldname`. For example:

```
nc cmd vovselect fieldname from jobs
```

Many types of objects in the vovserver may be queried. See the help information below for the supported objects.

 **Note:** `vovselect` supports the "\*" wildcard to signify all fields of a particular object. Be sure to quote the \* character as required by your shell, e.g.: `vovselect '*' from jobs where idint==12345`

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

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

`GRABBEDRESOURCES.<name>` only returns a value for the corresponding central resource when the job is currently running.

`SOLUTION.<name>` returns a value for the corresponding hardware resource after the job has started running. The value persists after the job has terminated.

`RESOURCES.<name>` attempts to determine a value for the corresponding requested resource. If the job is running, then the actual value of the allocated resource is returned. If the job is not running, the query will estimate a value by looking for the first matching value in the requested resource string. This may result in an underestimate, or an incorrect value.

For example, with the request: `-r "RAM/20 RAM/30"`, `RESOURCES.RAM` may return "20" or "50" depending on the scheduling phase of the job. A contrived example which illustrates the difficulty of computing a value would be `-r "( RAM/100 CLOCK/10 ) OR ( RAM/50 CLOCK/20 )"`. `RESOURCES.RAM` returns exactly the same value as `REQRAM`, and similarly for `CORES`, `PERCENT`, `SLOTS` & `SWAP`.

## Examples

Queries you can run include the following:

- `RESOURCES.<RESNAME>` estimates the requested resource value of `RESNAME`. It attempts to determine a value for the corresponding requested resource. If the job is running, then the actual value of the allocated resource is returned. If the job is not running, the query will estimate a value by looking for the first matching value in the requested resource string. This may result in an underestimate, or an incorrect value.

For example, with the request `-r "RAM/20 RAM/30"`, `RESOURCES.RAM` may return "20" or "50" depending on the scheduling phase of the job. A contrived example which illustrates the difficulty of computing a value would be `-r "( RAM/100 CLOCK/10) OR ( RAM/50 CLOCK/20 )"`. `RESOURCES.RAM` returns exactly the same value as `REQRAM`, and similarly for `CORES`, `PERCENT`, `SLOTS` & `SWAP`.

- `GRABBEDRESOURCES.<RESNAME>` returns the current value of `RESNAME` in a job's grabbed resources. It only returns a value for the corresponding central resource when the job is currently running.
- `SOLUTION.<RESNAME>` returns the value of `RESNAME` in a job's solution. It returns a value for the corresponding hardware resource after the job has started running. The value persists after the job has terminated.

For example, the following job submission

```
$ nc run -r "RAM/30 License:MATLAB/2" - sleep 1000
```

`vovselect` produces the following output:

```
$ nc cmd vovselect -header id,RESOURCES.RAM,RESOURCES.License:MATLAB from jobs
id RESOURCES.RAM RESOURCES.License:MATLAB
000001366 30 2
```

If the query is unable to determine a value, it will return an empty string.

If the query is used inside a 'where' clause, it may need to be quoted, such as, `-where '"grabbedresources.License:MATLAB">1'`.

For example, if the following job executes:

```
$ nc run -r License:MATLAB/2 - sleep 1000
```

vovselect produces the following output:

```
$ nc cmd vovselect -header id,statusnc,GRABBEDRESOURCES.License:MATLAB,SOLUTION.RAM
from jobs -where
'"GRABBEDRESOURCES.License:MATLAB">1'
id statusnc GRABBEDRESOURCES.License:MATLAB SOLUTION.RAM
000001202 Running 2 20
```

## vovselect

vovselect: Usage Message

Utility to query vovserver data.

USAGE:

```
vovselect <FIELDSPEC> from <OBJECT> [OPTIONS]
```

OPTIONS:

```
-h                -- Show usage syntax.
-v                -- Increase verbosity.
-where <FILTER>   -- Filter the results.
-order <COLUMN> [ORDER] -- Sort the output by the specified column
                        and ordering. Ordering is either "asc"
                        (ascending) or "desc" (descending).
                        Default ordering is ascending. When
                        specifying an ordering, place the column
                        and ordering in quotes, e.g.
                        -order 'name desc'.
-limit <N>        -- Limit the output to N rows.
-distinct         -- Return distinct rows.

-header           -- Displays column headers in the output.
-cache 0/1        -- Control cacheing of query (default is 1).
                    Tech Note: use cache 0 for small results
                    (less than a few thousand rows)
```

If option values contain shell-sensitive characters, such as ">", enclose them with single quotes (Linux) or double quotes (Windows).

The from parameter will accept queryable object names (as listed by "vovselect objectname from objects"), individual object identifiers (as listed by "vovselect idint from <object>"), or set names (as listed by "vovselect name from sets"). This parameter can also accept the following:

```
SUBSETS.<SETID>
MATCHES.<RESMAPID>
MATCHES.<RESMAPNAME>
```

SUPPORTED OBJECTS:

Run "vovselect objectname from objects" to see the list of queryable objects.

SUPPORTED FIELDS:

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



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

#### SUPPORTED FILTERS:

Use selection rule operators in conjunction with field names to filter queries. See operator list at <URL/doc/html/vov/topics/vov/operators.htm> via web browser. To get the current the URL for current instance, execute the vovbrowser command.

#### EXAMPLES:

```
% vovselect -h
% vovselect objectname from objects
% vovselect fieldname from server
% vovselect id,name from users -order name -limit 10 -header
% vovselect id,name from users -where 'name==joe'
% vovselect id,name from 12345
% vovselect id,name from subsets.23456
% vovselect matchtype,host from matches.License:spice
% vovselect idint,name from users -where 'idint>3600'
    -order 'idint desc'
% vovselect id,age from System:running
% vovselect id,age -from System:running -cache 0
```

## Use vovselect for Querying

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

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

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

nc hosts	vovselect from TASKERS	vovselect from HOSTS
ARCH	ARCH	ARCH
CAPABILITIES	CAPABILITIES	NA
CAPACITY	CAPACITY	CPUS
CLASSRESOURCES	CLASSRESOURCES	NA
CLOCK	CLOCK	CPUCLOCK

<b>nc hosts</b>	<b>vovselect from TASKERS</b>	<b>vovselect from HOSTS</b>
COEFF	COEFF	NA
CONSUMABLES	CONSUMABLES	NA
CORES	CORESAVAIL	NA
CORESAVAIL	CORESAVAIL	NA
CORESTOTAL	CORESTOTAL	CPUS
CORESUSED	CORESUSED	NA
CPUS	CPUS	CPUS
CURLOAD	CURLOAD	NA
DOEXEC	DOEXEC	NA
DONETINFO	DONETINFO	NA
DOPROCINFO	DOPROCINFO	NA
DORTTRACING	DORTTRACING	NA
EFFLOAD	NA	NA
EXTRAS	EXTRAS	NA
FULLINFO	FULLINFO	NA
GROUP	GROUP	NA
HB	NA	NA
HBPP	NA	NA
HEARTBEAT	HEARTBEAT	NA
HOST	HOST	NAME
ID	ID	NA
IDINT	IDINT	NA
LASTJOBID	NA	NA
LASTUPDATE	LASTUPDATE	NA
LIFETIMEJOBS	LIFETIMEJOBS	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
LOAD1	NA	NA
LOAD15	NA	NA
LOAD5	NA	NA
LOADEFF	NA	NA
MACHINE	MACHINE	MACHINE
MANUALPOWER	NA	NA
MAXLOAD	MAXLOAD	NA
MESSAGE	MESSAGE	NA
MESSAGESYS	MESSAGESYS	NA
MESSAGEUSER	MESSAGEUSER	NA
MODEL	MODEL	NA
NAME	NAME	NAME
NUMJOBS	NA	NA
OSCLASS	OSCLASS	NA
PERCENT	PERCENT	NA
PERSISTENT	PERSISTENT	NA
PID	PID	NA
POWER	POWER	NA
RAM	RAM	NA
RAMFREE	RAMFREE	NA
RAMTOTAL	RAMTOTAL	RAMTOTAL
RAWPOWER	NA	NA
RELEASE	RELEASE	NA
RESERVEDBY	RESERVEDBY	NA
RESERVEEND	RESERVEEND	NA

<b>nc hosts</b>	<b>vovselect from TASKERS</b>	<b>vovselect from HOSTS</b>
RESERVEFORBUCKETID	RESERVEFORBUCKETID	NA
RESERVEFORID	RESERVEFORID	NA
RESERVEGROUP	RESERVEGROUP	NA
RESERVEJOBCLASS	RESERVEJOBCLASS	NA
RESERVEJOBPROJ	RESERVEJOBPROJ	NA
RESERVEOSGROUP	RESERVEOSGROUP	NA
RESERVESTART	RESERVESTART	NA
RESERVEUSER	RESERVEUSER	NA
RESOURCECMD	RESOURCECMD	NA
RESOURCES	NA	NA
RESOURCESEXTRA	NA	NA
RESOURCESPEC	RESOURCESPEC	NA
RUNNINGJOBS	RUNNINGJOBS	NA
SLOTS	NA	NA
SLOTSTOTAL	SLOTSTOTAL	NA
STATSREJECTCORES	STATSREJECTCORES	NA
STATSREJECTOTHER	STATSREJECTOTHER	NA
STATSREJECTRAM	STATSREJECTRAM	NA
STATSREJECTRESERVED	STATSREJECTRESERVED	NA
STATSREJECTSLOTS	STATSREJECTSLOTS	NA
STATSVISITS	NA	NA
STATUS	NA	NA
SWAP	SWAP	NA
SWAPFREE	SWAPFREE	NA
SWAPTOTAL	SWAPTOTAL	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
TASKERGROUP	TASKER	NA
TASKERNAME	TASKERNAME	NAME
TASKERSLOTSSUSPENDABLE	TASKERSLOTSSUSPENDABLE	NA
TASKERSLOTSSUSPENDED	TASKERSLOTSSUSPENDED	NA
TASKERSLOTSUSED	TASKERSLOTSUSED	NA
TASKERTYPE	TASKERTYPE	NA
TIMELEFT	TIMELEFT	NA
TMP	TMP	NA
TYPE	TYPE	NA
UPTIME	NA	NA
UPTIMEPP	UPTIMEPP	NA
USER	USER	NA
VERSION	VERSION	NA
VOVVERSION	VOVVERSION	NA

## Invalidate Dependent Nodes

The utility `vovtouch` toggles the status of the given nodes twice, so that each node passes through a state of being `INVALID`. The end state of each node depends on what state it started in.

The effect is that all dependent nodes are invalidated. This utility is generally used with [primary inputs](#). This is commonly done to force a retrace of a portion of the design.

Unlike the UNIX utility `touch`, `vovtouch` does not change the timestamp of the files.

### vovtouch

Toggles the status of the given nodes twice, so that each node passes through a state of being `INVALID`.

```
vovtouch: Usage Message
```

DESCRIPTION:

Toggle the status of nodes twice following one of these sequences:

Start				Finish
-----				
VALID	->	INVALID	->	VALID
INVALID	->	VALID	->	INVALID
FAILED	->	VALID	->	INVALID

Nodes with status different from VALID, INVALID or FAILED are not affected.

This has the effect of invalidating all dependent nodes.

USAGE:

```
% vovtouch <node_descriptor> ...
```

where the node descriptor is either the name of a file or the VovId of a node in the dependency graph.

OPTIONS:

```
-h
-help          -- This message.
-v            -- Increase verbosity.
```

EXAMPLES:

```
% vovtouch aa
% vovtouch 00023456 00023457
```

## Comments

You can produce a similar effect of invalidating all dependent nodes using the command `vovinvalidate`.

The difference is that with `vovinvalidate`, the target nodes are changed to the INVALID state, while with `vovtouch`, the target nodes may end up back in a VALID state if it was in a VALID state to begin with.

## vovinvalidate

Invalidate nodes in the dependency graph. The objects are typically files or jobs, which are passed by VovId or by file name, and sets, which are passed by name.

vovinvalidate: Usage Message

Invalidate nodes in the dependency graph.  
The objects are typically files or jobs, which are  
are passed by VovId or by file name, and sets, which are  
passed by name.

USAGE:

```
vovinvalidate [OPTIONS] <nodeId> ...
```

```
OPTIONS:
  -alljobs           - Invalidate all jobs.
  -allnodes          - Same as -alljobs (backwards compatibility).
  -dir <dirname> ... - Invalidate jobs in given directories.
                    Repeatable.
  -force             - Use the _FORCE suffix in vtk_node_change_status.
  -help             - Help usage message.
  -h
  [-hier] -set <setName> ... - Invalidate given set by name. Repeatable.
                             Invalidate all subsets also with -hier.
  -sicktasker        - Clears WHY and crash recovery-related
                             properties (for use with running/retracing
                             jobs that are no longer attached to a tasker).
  -silent            - Do not print count of invalidated nodes.
  -subdirs <dirname> ... - Same as -dir but recursive. Repeatable.
  -v                 - Increase verbosity
  -why <text_reason> ... - Optional reason for invalidation.
  --                 - End value list for an option.
```

#### EXAMPLES:

```
% vovinvalidate filea fileb filec
% vovinvalidate 00023456
% vovinvalidate -sicktasker 34567
% vovinvalidate -dir .
% vovinvalidate -set set1 set2 -- filea fileb 10332 10334
% vovinvalidate -alljobs -why "Doing a clean build"
```

## Clean Directories

The tool `vovcleandir` is used to delete unused files.

`vovcleandir` analyzes the directories passed as arguments and then enters an interactive loop in which you can choose to:

- list all files to be removed
- remove all or only selected files
- quit

With the option `-force`, `vovcleandir` removes all unused files without asking for further input.

```
% vovcleandir -help
% vovcleandir
% vovcleandir -force
% vovcleandir *.vhd1
```

With the option `-all`, `vovcleandir` checks all known directories for unused stdout and stderr files.

```
% vovcleandir -all
```

## vovcleandir

Clean up directories of all UNKNOWN files.

```
vovcleandir: Usage Message

DESCRIPTION:
  Clean-up directories of all UNKNOWN files.

USAGE:
  vovcleandir [options] [file_or_directory] ...

OPTIONS:
  -all           -- Cleanup all directories in which jobs are run.
  -force         -- Cleanup (delete) UNKNOWN files without asking
                  for confirmation.
  -h -help       -- Help usage message.
  -silent        -- Do not print the list of the deleted files.

  If you do not indicate any file, this utility tests all ./std.*
  and all vovstd/* files.

  Each file_or_directory argument is expanded according
  to glob rules.

EXAMPLES:
  % vovcleandir -help
  % vovcleandir .
  % vovcleandir -force
  % vovcleandir -all
  % vovcleandir *.vhd1

SEE ALSO:
  vovcleanup     -- to cleanup old log files from host and project
  nc clean       -- to cleanup old files generated by Accelerator
```

## Remove Nodes From the Graph

Forgetting nodes means eliminating them from the flow.

Forgetting a file from the flow does not remove the file itself from the filesystem. It only tells FlowTracer to stop paying attention to the file. Consider `vovblast` to simultaneously forget and delete a file. A job that is RUNNING or RETRACING cannot be forgotten: you need to [stop](#) the job first.

In Accelerator, jobs can be forgotten automatically by setting the "autoForget" flag. The log files are also automatically forgotten. They are removed from the disk if their path contains the string "vnc\_logs".

### Forget Nodes using the CLI

The program to forget nodes from the flow is `vovforget`.



vovforget: Usage Message

DESCRIPTION:

Forget objects from the project's vovserver.  
The objects are typically nodes (files or jobs) and are  
are passed by VovId or by file name.  
The objects may also be users, sets, or selected by option.

USAGE:

vovforget [options] [objectId] ...

OPTIONS:

-alerts -- Forget all alerts.  
-allalerts -- Forget all alerts (same as -alerts).  
-allemptysets -- Forget all empty sets older than 60s.  
(Command is only supported for Accelerator, Accelerator

Plus, and Hero products.)

-allnodes -- Forget all nodes.  
-allhosts -- Forget all hosts.  
-allsets -- Forget all sets (except System sets).  
see also -allemptysets and -emptysets  
(Command is only supported for FlowTracer, Allocator, and

Monitor products.)

-allresources -- Forget all resource maps.  
-rdsresources -- Forget all resource maps created by RDS.  
-allreservations -- Forget all reservations.  
-blocking -- Forget blocking files, i.e.  
files with no inputs that are not VALID.  
-dir <dirname> -- Forget jobs in a directory.  
-elements <setName> -- Forget elements in given set.  
-emptysets -- Forget 'my' empty sets older than X seconds  
X = 0s for FlowTracer, Allocator, and Monitor products.  
X = 60s for Accelerator, Accelerator Plus, and Hero

products.

-emptysets -hier -- Forget 'my' empty sets (and their subsets) older than 0s  
(default).  
(Command is only supported for FlowTracer, Allocator, and

Monitor products.)

-h  
-help -- Help usage message.  
-hier -- When forgetting a set or its elements,  
also traverse the hierarchy of the subsets.  
Technical note: The traversal is based on  
attachments of subsets,  
and not on the names of the subsets.  
-idlelimits -- Forget all Limit:\* resources that are not  
associated with any running or queued jobs.  
-idleusers -- Forget users with no clients and no jobs.  
-isolated -- Forget isolated files, i.e.  
files with no inputs nor outputs.  
-missing -- Forget missing files.  
-orphanreservations -- Forget all orphan reservations.  
-oldlimits -- Forget all Limit:\* resources that have not  
not been used in the last 7 days.  
-preemptrules -- Forget all preemption rules.  
-q  
-s  
-set <setName> -- Forget a set.  
-silent -- Do not print count of forgotten nodes.  
-sleeping -- Forget all nodes with status SLEEPING  
(they appear in black in the GUI).  
-subdirs <dirname> -- Forget jobs in a directory and all subdirs.

```
-v                -- Increase verbosity.

EXAMPLES:
% vovforget aa bb cc                -- Forget files by name.
% vovforget 00023456              -- Forget objects by id.
% vovforget -dir .                -- Forget jobs in current dir.
% vovforget -subdirs .            -- Forget jobs in current dir
                                   and all subdirs.

% vovforget -allnodes
% vovforget -blocking
% vovforget -allemptysets          -- All empty sets, for all users.
% vovforget -emptysets            -- My empty sets
% vovforget -emptysets -age 10s    -- My empty sets older than 10s

% vovforget -hier -set FrontEnd
% vovforget -hier -elements FrontEnd

SEE ALSO:
vsz                                -- To forget isolated nodes.
vovset forget <set>               -- To forget sets.
```

- To forget specific files or jobs, give either the name of a file or the VovId of a job as an argument.  
Examples:

```
% vovforget fileAA
% vovforget 00012345 00023456
```

- To forget **all elements in a set**, use the -elements option. Example:

```
% vovforget -elements "failed:jobs"
```

- To forget all **jobs in a directory**, use the -dir option. Example:

```
% vovforget -dir .
```

- To forget all **jobs in a directory and all of its subdirectories**, use the -subdirs option. Example:

```
% vovforget -subdirs .
```

- To forget all **blocking files**, use:

```
% vovforget -blocking
```


## Forget Nodes

### Forget Nodes using the GUI

To forget a single node, visualize it in the Set Viewer, right click on it and click **Forget**.

To forget many nodes, [select](#) them and then forget them using the pop-up menu.

### Forget Nodes using the Browser Interface

The browser interface presents several entries to forget individual nodes or elements of a set. The link is normally associated with the icon . Confirmation will be requested.

## Forget Nodes and Sets using the Tcl Interface

- Forget a single node using `vtk_node_forget $nodeId`
- Forget all nodes in a set with `vtk_set_forget -elements $setId`
- Forget a set (not its elements) with `vtk_set_forget $setId`

## Remove Files

To remove a file from the file system which is currently registered in a flow graph, you should notify FlowTracer that the file is going away. The simple way to do this is to use the `vovblast` command.

This command directs FlowTracer to adjust the flow graph to reflect that the file is going away, and to remove the file for you.

1. It **invalidates all nodes** dependent on the files.
2. It **forgets files** from the flow.
3. It removes the files from the disk.

By using `vovblast` instead of `rm`, you maintain flow consistency.

Using this command to remove files is called *blasting the files*.

```
vovblast: Usage Message

DESCRIPTION:
  Forget files from the trace AND DELETE THEM from the disk.
  The files must be specified by name.  Each file argument
  is interpreted as a regular expression.
  All dependent nodes are invalidated.

USAGE:
  % vovblast [options] [file] ...

OPTIONS:
  -help           -- This message.
  -silent         -- Be quiet.

  Options can be abbreviated to one character.

EXAMPLES:
  % vovblast a b c
```

Example:

```
% vovblast aa
vovblast: message: Forgetting aa
vovblast: message: Removing aa
```

## Remove Large Files That Are Easy to Recreate

The tool `vovremove` offers advanced management of file system disk space

Suggestion of file to delete are based on the:

- Role of the file in the design flow
- Size of the file
- Age of the file
- Time it takes to regenerate the file

It is better to delete large files that take little time to generate rather than small files that take a long time to generate. Also, it does not make sense to delete a file before all the jobs that use it have been completed successfully.

Each file that is a candidate to be deleted is assigned a "grade," defined as the logarithm of the ratio between the file size and the time it has taken to generate the file. Files with grade lower than a given threshold will not be deleted. If you want to free up more disk space, lower the grade threshold.

Example:

```
% vovremove -rx '\.obj$'
vovremove: message: Rule is 'isfile STATUS==VALID HASINPUTS NAME~\.obj$'
vovremove: message: Getting data ...
vovremove: message: Analyzing 1219 files...
Files 1000 ...
vovremove: message: Found 1219 candidates
vovremove: message: Exploring space/time tradeoff...
```

Label:	Grade	MaxJobCost	MinFileAge	-->	#Files	TotSize	TotTime
1:	6.0	1m00s	3d00h	-->	0		
2:	5.0	1m00s	3d00h	-->	0		
3:	4.0	1m00s	3d00h	-->	4	114kB	10s
4:	3.0	1m00s	3d00h	-->	652	16MB	1h58m
5:	2.0	1m00s	3d00h	-->	1174	19MB	3h48m
6:	2.0	3m00s	3d00h	-->	1180	19MB	3h56m
7:	1.0	3m00s	3d00h	-->	1218	19MB	4h16m
8:	1.0	10m00s	3d00h	-->	1219	19MB	4h25m
MAX:	0.0	100d00h	0s	-->	1219	19MB	4h25m

Select a label >

Given the destructive nature of the operation, the tool requires the explicit option `-doit`.

### vovremove

Remove files intelligently, considering the role of the file in the design flow, the size of the file, and the time it takes to generate the file. This operation works only for data in the FILE database.

```
vovremove: Usage Message
DESCRIPTION:
  Remove files intelligently, considering
  1- the role of the file in the design flow,
  2- the size of the file,
```

3- the time it takes to generate the file.

This operation works only for data in the FILE database.

USAGE:

vovremove [options]

OPTIONS:

```
-everywhere    -- Look everywhere for files to delete.
-dir <dir>     -- Select a directory under which are the files
               you want to be deleted.
-rx <rx>       -- A regular expression to select files.
-age <timespec> -- Minimum age of file to be deleted.
-D <timespec>  -- Maximum cost of recomputation of file.
               Files that take longer to compute will not be removed.
-G <number>    -- Minimum 'Grade' of files to delete:
               Grade <= log10( size_of_file / time_to_recompute )
-doit          -- Really do it, otherwise, just show.
-help          -- This message.
```

EXAMPLES:

```
% vovremove -h
% vovremove -dir .
% vovremove -rx '\.obj$' -doit
```

# Graphical User Interface

## The GUI Console

The graphical user interface, called the "console", is invoked with the command `vovconsole`.

Since `vovconsole` displays the state of a specific FlowTracer project, you must choose which one by using `vovproject enable project-name` before starting the GUI.

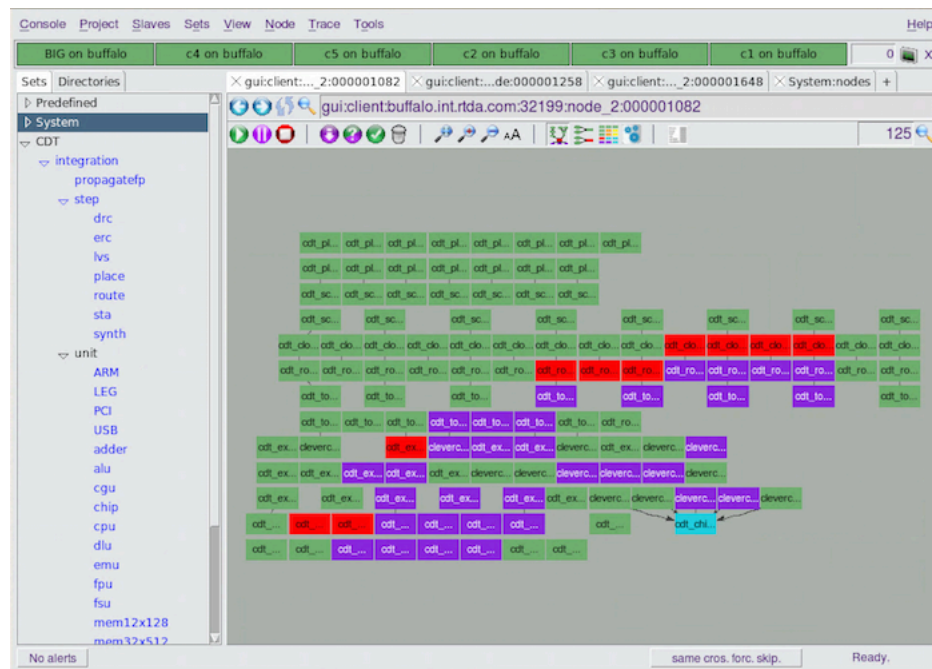


Figure 19:

The console consists of four panels:

- The Set Browser is the panel on the left. It displays sets defined in the project. This Directories panel also operates as a specialized directory browser.
- The Set Viewer is the panel on the right. It displays the elements in the current set in one of three possible views: horizontal graph, vertical graph, or grid. It can also display summary information about the current set.
- The System Log is across the bottom of the window. It is hidden by default. It can be shown with the menu option **Console > Show/Hide > Show System Log**. It lists all important events.
- The Graphical Tasker Monitor is across the top of the window. It shows the state of the tasker hosts.

In addition to these panels, there is a navigation menu at the top, and two rows of action buttons below that. There is also a search field for selecting nodes by name.

vovconsole: Usage Message

```
USAGE:
    vovconsole [options]
OPTIONS:
    -batch <file>          -- Execute specified file after the console is
                           ready.
    -fontsize <size>       -- Specify a size for the normal font.
                           Default is 10.
    -geometry WxH          -- Specify initial size of console.
    -help                  -- Get this message.
    -initsystemnodes       -- Initialize the console upon start-up with
                           all nodes in the trace.
    -monochrome            -- Use gray except for top bar.
    -proj <vovproject>     -- Does nothing. Used when bsub'ing the
                           vovconsole to track which console is on
                           which project.
                           See: VOVCONSOLE_SUBMIT_CMD

    -set <setNameOrId>     -- Show specified set in a new pane using the
                           view specified with the previous -view option.
                           Default to not to show any set.
                           May be repeated.
    -showcustomize         -- Show areas that can be customized.
    -v                     -- Increase verbosity.
    -view grid|graph|stat  -- Specify the initial view for the set
                           specified with -set.
                           Default is 'grid'

EXAMPLES:
    % vovconsole &
    % vovconsole -fontsize 12 &
    % vovconsole -geometry 1000x500 &
    % vovconsole -view vertical -set System:jobs -view grid
    % vovconsole -set System:files &
    % vovconsole -showcustomize
```

## Set Browser

The Set Browser allows you to list and edit the sets defined in the graph.

A *set* is a named collection of nodes. Some sets are managed by the system, others are based on [Selection Rules](#), and still others are created as a side effect of certain operations such as starting a retrace.

In the Set Browser panel, the sets are organized into 4 classes:

### *Predefined*

Predefined sets, all of which are based on selection rules. These sets are most commonly used in the interaction with FlowTracer. For example, the set "Stuff to do" consists of all the jobs that are not VALID, while the set "Failed jobs" consists of all the jobs with status equal to FAILED. Initially these sets do not exist and show as black color. When one of these set names is double-clicked, the real set is created and the color turns blue.

### System

System sets are internal to the server. They are read-only sets. Among the most useful system sets you have: "All nodes" "All jobs" "All files". There is no selection rule associated with system sets.

### Temporary

Temporary sets are generated by activities such as starting a retrace and viewing a set of nodes.

### User

User defined sets are all the other sets. They are listed in alphabetical order and grouped by the first letter of the set name.

## Select a Set

1. To select a set, double-click on its name.  
The set is recomputed and up-to-date set is drawn on the Set Viewer.
2. Predefined sets are defined on vovconsole for convenience. The sets can be deleted or modified. Vovconsole will keep the set names as disabled when they are deleted. Double-click a predefined set.  
The sets are created again.
3. If the set is based on [Selection Rules](#), you can use the **Refresh Set** button to update the set display, meaning that the nodes in the set are recomputed on the basis of the selection rule.

## Create a New Set

1. To create a new set, click **Sets** > **New** to open the dialog shown here.



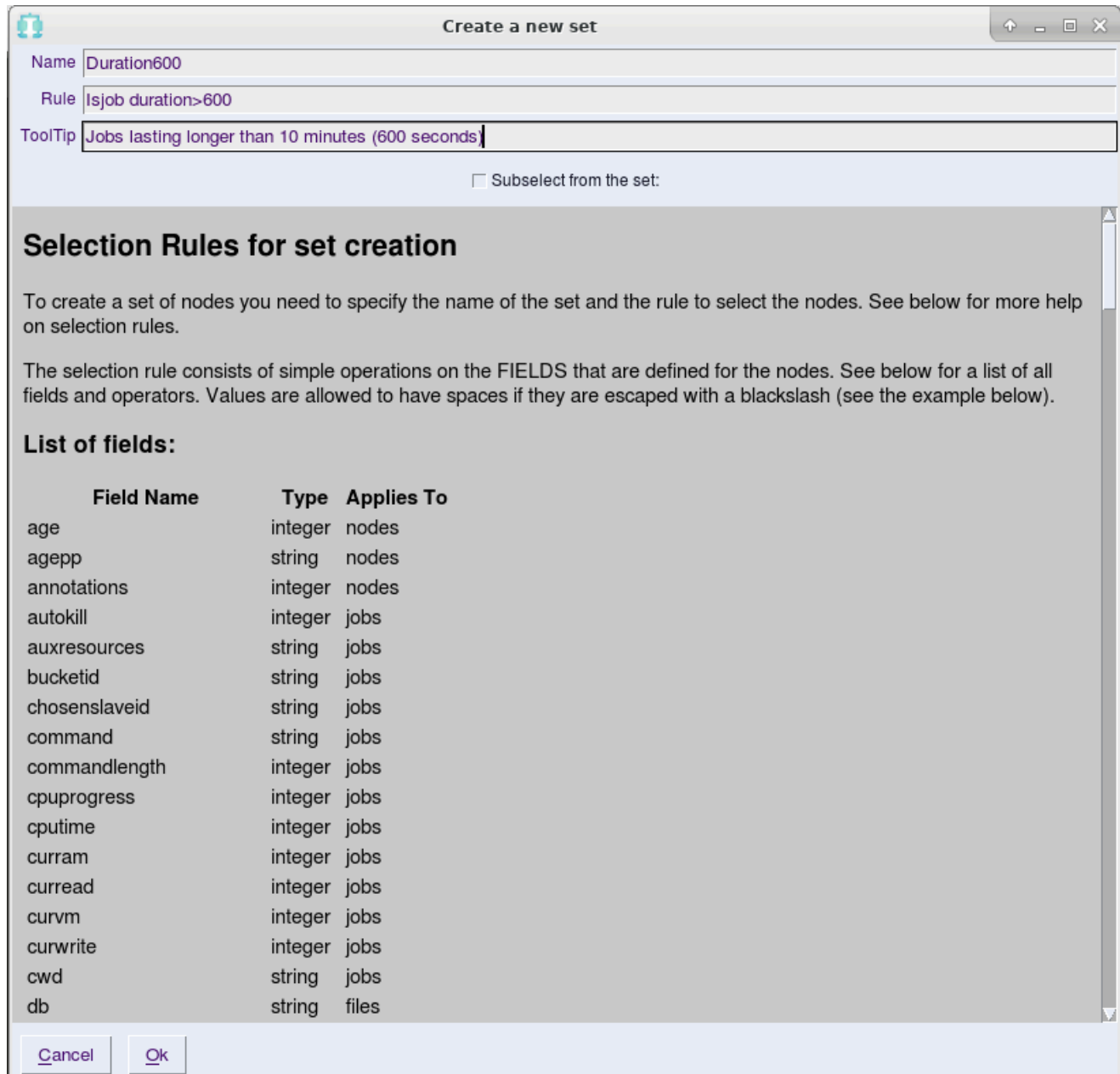


Figure 20:

2. Specify a **name** and a selection rule.
3. To have the new set be a subset of the currently selected set, click **Subselect from**.
4. Click on the **Directory** tab.  
The Set Browser turns into a specialized Directory Browser. This allows you to browse the directories of your design and to create temporary sets on the fly consisting of jobs and files contained in or below a particular subdirectory. Single-click on the right arrow symbol to expand a directory and on the down arrow symbol to collapse it. Double-click on the directory name to create or update the set of all jobs and files contained in that directory.

## Forget a Set

Forgetting a set does not affect the existence of its elements. Only the container is gone, not the contents.

 **Note:** You cannot forget the system sets.

1. Select the sets to forget by clicking on them (use Ctrl-click and Shift-click to extend the selection) and then click **Sets > Forget**.
2. You can forget the predefined sets, but the hierarchical element for that set still remains in the Set Browser. Double-clicking on that element will recreate the corresponding set.

## Set Viewer

When you select a set by double clicking on it in the Set Browser panel, you display that set in the Set Viewer panel.

You can choose between four different views of that set:

- The Vertical Graph View displays the nodes in the set as they appear in the dependency graph, complete with arcs, with the earliest nodes in the dependency at the top.
- The Horizontal Graph View displays the nodes in the set as they appear in the dependency graph, complete with arcs, with the earliest nodes in the dependency graph on the left.
- The Grid View is a compact graphical representation that shows a colored square for each job and a colored circle for each file. Jobs and files are placed on a regular grid. The advantage of this view is the ability to display a large number of elements in a relatively small window.
- The Stat View is a summary page showing textual information about the set.

In the three views showing node, you can select nodes with the mouse and invoke a pop-up menu with the right button. This menu allows you to perform operations on a single node, such as editing, invalidating, and forgetting.

Double-clicking on a node brings up the **Node Editor** for that node.

A "rubber band" operation (point, click, hold, drag, and release) selects all the nodes that have any pixels within the rubber band area. When you make a selection, the selected nodes are highlighted. If you pop-up the menu while pointing to a node in the selection, the menu operations affect all the nodes in the selection.

## Graphical Tasker LED Monitor

The top of the console is the graphical taskers LED monitor. It consists of a grid of sunken rectangles, each representing a tasker host. The color of the rectangle indicates the tasker status. In configurations with lots of taskers, each rectangle gets small, reduced to almost like a status LED on a phone.

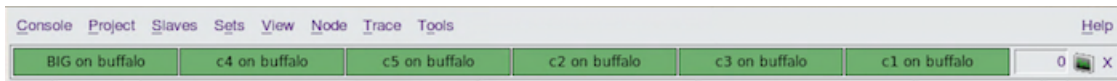


Figure 21:

Right-clicking on rectangle activates a pop-up menu to control the taskers and the job currently being executed by the taskers.

The taskers LED monitor has limited space, so it can only show a few taskers. If there are more taskers than can fit in the space of the LED, they get automatically grouped together by their status. When grouped like this, a single LED is shown for each group of taskers with similar status. The number in the label indicates how many taskers are in that group.

You can choose to group taskers by a different criteria by selecting the criteria in the dropdown next to the LEDs. Possible grouping criteria are: Arch, Status, and Group. Alternately, you can choose to see all taskers which will then display an LED for each tasker without any grouping.



**Note:** When displaying all taskers, there may not be enough space to display all taskers. To see all the taskers, you can display the floating taskers monitor by choosing the **Console > Floating Taskerlists Monitor** menu item.

## System Log

The system log, located at the bottom of the GUI, shows relevant events, one per line.

Look at this window for feedback on the various operations performed through the GUI.

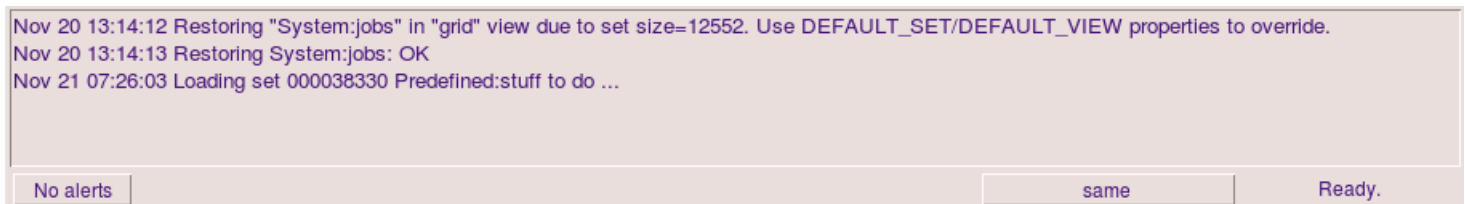


Figure 22:

The system log window also doubles up as a Tcl interpreter. You can enter one line commands in the window and evaluate them by pressing Enter.

## Customize the GUI

### Console Window Size

You can adjust the minimum size of the window through the **Window Size** tab of the **Preferences** dialog.

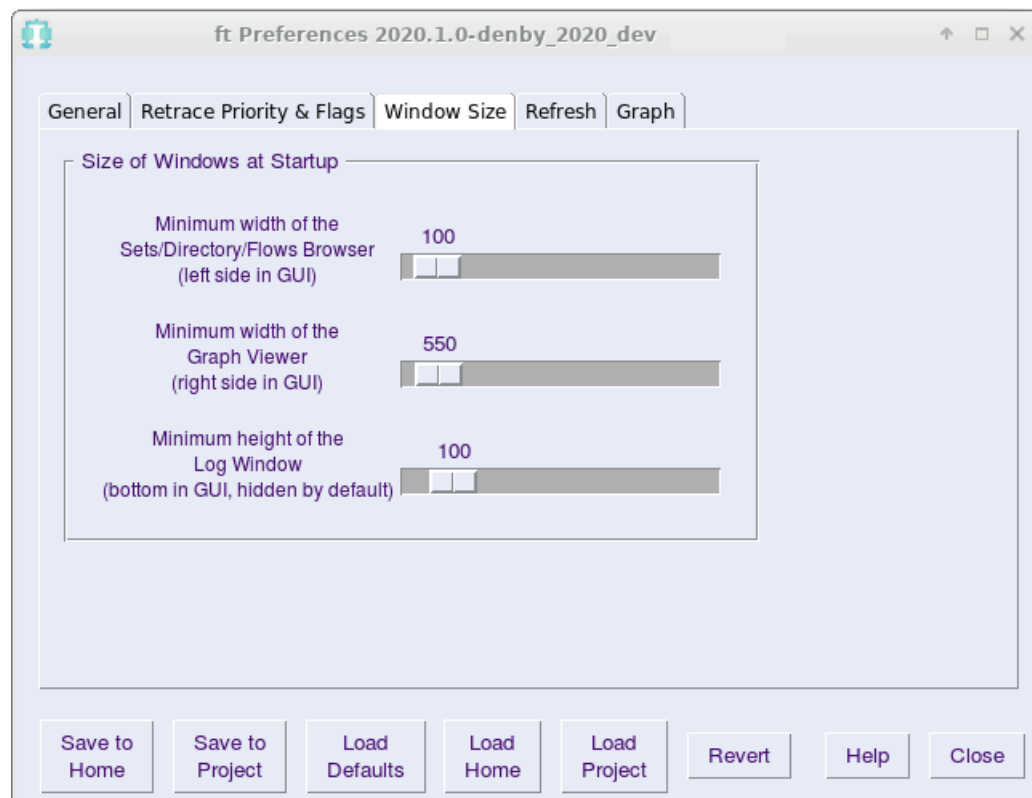


Figure 23:

## Customize Component Graphs

You can also adjust the size and graphing of the components through the **Graph** tab of the Preferences dialog as shown below. The adjustments include the type of connectors for connecting the nodes, node spacing, whether or not to "break" long rows, and more.

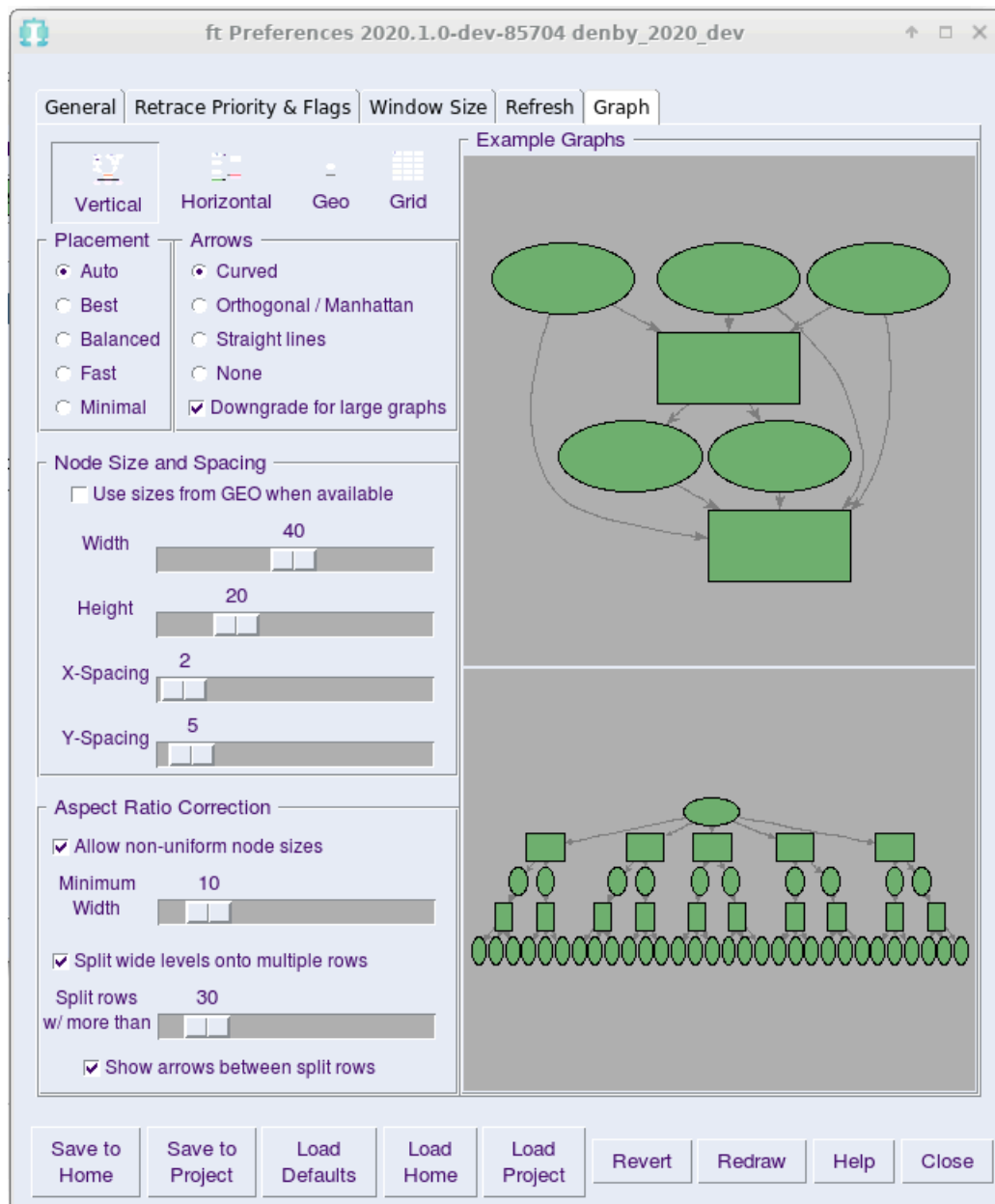













Figure 24:










### General Customization with vovautoexec.tcl

If a file named `vovautoexec.tcl` is present in the directory from which `vovconsole` is started, it is sourced when starting the console. You will need to study the `vovconsole` code and have Tcl/Tk programming skills to make effective use of this method.

## Icons

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

	Run/Rerun	Update the current object, whether a set or a node. All jobs necessary to bring the object up to date are scheduled for execution. The jobs that can run now start running.
	Stop/Dequeue	Un-schedule a job that is not yet running or abort a retrace request. Running jobs are unharmed and keep running. Graceful Stop. Cyan/asked-to-run jobs will turn back purple/invalid.
	Stop/Dequeue	Stop a running job or a retrace request. This does the same thing as the dequeue above, but in addition kills the selected running jobs. It's a forced stop. Running jobs that have been stopped will turn red.
	Navigate Backward	Display the previous set that you were browsing.
	Navigate Forward	Display the next set that you were browsing, when you already went backward.
	Refresh	Recompute the current set based on the selection rule.
	Find	Finds files or jobs in the trace. There are two identical Find icons. The left one close to the name of the set being displayed triggers changing the setname box into a search box for the current graph. The other on the extreme right brings up a new dialog that lets you search for files or jobs.
	Invalidate	Invalidates the selected nodes.
	Try Validate	Tries to validate the selected jobs and its downcone.
	Force Validate	Equivalent of <code>make -t</code> .
	Forget	Removes the job/file from the graph of dependencies. This does NOT remove the file from disk. It just removes it from the dependencies allowing blocked jobs to start. Usually rerunning

		vovbuild is enough to bring the dependency back.
	Fit	Reduces/expands the graph so that it fits in the window.
	Zoom In	Expands the graph to better see some nodes.
	Zoom Out	Reduces the graph to see more nodes.
	Select Font Size	Reduce or enlarge the size of the characters in the nodes.
	Vertical View	Display the current set using the "graph" widget.
	Horizontal View	Display the current set using the "horizontal" widget.
	Grid View	Display the current set using the "grid" widget.
	Stats View	Display the current set using the "stats" widget.
	Graph Settings	Set the various preferences for Weight Driven Placement. This takes you to the <b>Graph</b> tab of the Preferences dialog.


## Using the Set Browser and Set Viewer

The GUI console provides a Set Browser navigation panel on the left side, and a Set Viewer panel on the right side.

### Navigation of Sets



You can interact with the Set Browser navigation by single clicking on set choices and expand/contract controls. This single clicking interaction only affects the Set Browser navigation panel and the context within that navigation. You are only interacting with that navigation control panel. You must double click on a specific set to cause the Set Viewer panel to make that set the subject of its view.

When you double click on a given set, that set becomes the subject of the Set Viewer panel.

 **Note:** You must click the **refresh** button to cause the Set Viewer panel display to be updated to show the current and full report about nodes in the set, based on the selection rules for the set. The refresh uses the set selection rules to identify all the nodes that currently match the rules to belong to the set.

The mode of display showing the nodes in the subject set can be modified by selecting a particular view of the set. You can view the detailed nodes in the set in three different ways, or you can view a summary report on attributes of the set. When viewing the nodes in the set, you can view them as a dependency graph from two different perspectives, showing the nodes and their dependency arcs, or you can just simply view the nodes without any information about their dependency.

### Move Backwards and Forwards to Viewed Sets

As you click on sets in the Set Browser to view them in the Set Viewer, you can move back to the previous set you had in view by clicking on the **Backwards** icon  in the upper left, in the command row. After moving backwards, you can then move forward by clicking **Forwards** icon .

At any time you can hold down the left mouse click on either the **Backwards** or **Forwards** icon to pop-up a navigation list of sets visited. You can click on any of the sets in the list to switch to making that set the subject of the Set Viewer.

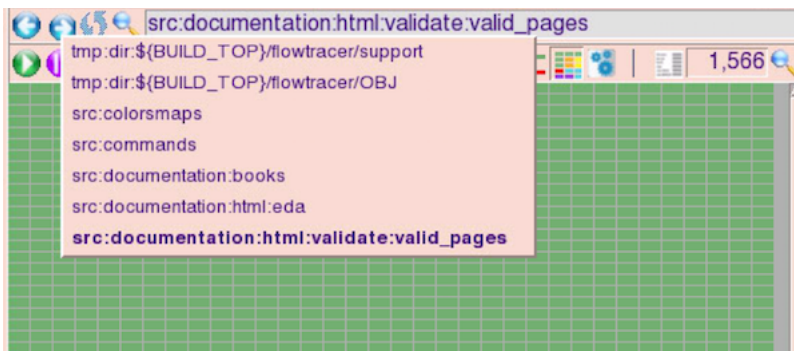


Figure 25:

### Vertical Graph View of a Set

This view displays the nodes in the set as a graph showing their dependency relationship as arcs between the nodes. The graphical relationship has a vertical perspective. The earliest dependencies in the graph are shown at the top of the graph, with latest dependencies shown at the bottom of the graph.



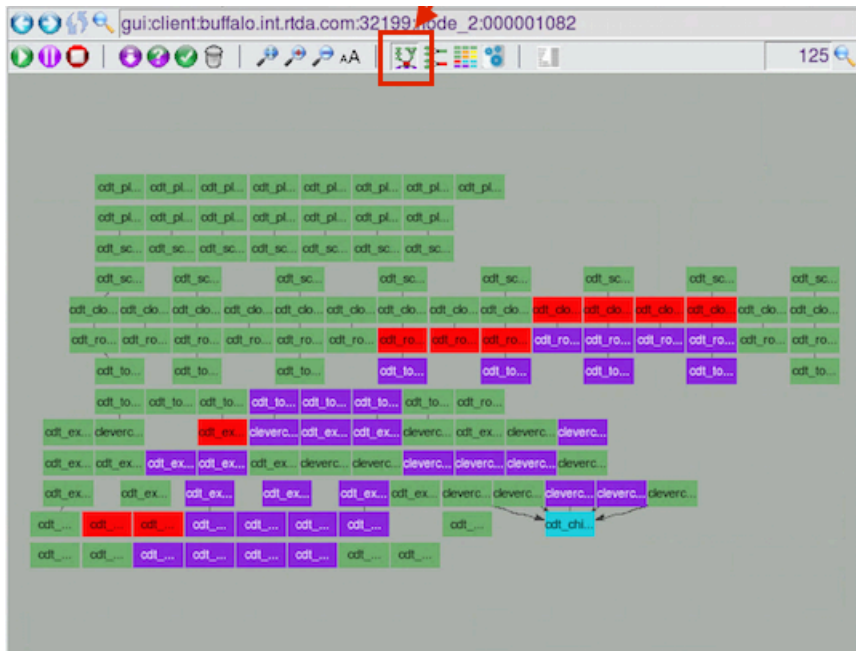


Figure 26:

Open the Vertical Graph view by clicking the **Vertical Graph** icon  or by using the keyboard shortcut **shift-G**.

In the Set Viewer:

- Hover the mouse over a node; a pop-up dialog appears that describes that node.
- Left-clicking on a node selects it. A yellow border around the node indicates that it is selected.
- Right-clicking on a node displays a context menu. This menu provides options so you can perform selected operations on that node.
- Double-clicking the left button on a node opens the Node Editor in a separate window to show the properties of the current node.
- When the Node Editor dialog opens, it provides a set of tabbed pages. The default view for Job Nodes (rectangles) is the **Job Info** tab. The default view for File Nodes (circles) is the **File** tab.

Altair FlowTracer Node Editor 2019.01

000039588 **VALID** vov ./compute\_sizes ../../../../mkcdrom/cdrom pkg packages.tcl

Job Info Execution & Impact Annotations & Properties History Why Stdout +

Flags **skip** autoforget preemptable profile

Host denby1 User.Group denby.rtda

Slave localhost Fair Share Group /time/users.denby

Job Name Job Class Job Proj

Environment BASE

Directory /home/work/hourglass/dir/

Command vov ./compute\_sizes ../../../../mkcdrom/cdrom pkg packages.tcl

Resources unix

Submitted	2019/11/21 07:51:37	Total Duration	7s	In Queue	8m45s	Max Ram	9 MB
Started	2019/11/21 08:00:22	Run Duration	7s	Exit Code	0	Max Swap	152 MB
Ended	2019/11/21 08:00:29	Suspended Duration	0s	Legal Codes	0	Job Placement	fastest
Expected	2019/11/21 08:00:29	Expected Duration	7s	Auto Kill	0s	Bucket ID	003202235

Notes  
PROP s ALLPIDS=74116  
PROP S SOLUTION=OSCLASS=unix RAM/20 CORES/1 SLOTS/1 PERCENT/1 TIMELEFT#8

Figure 27:

Choose the **Execution & Impact** tab for a Job Node to see a report on the downstream impact of this job running.

Altair FlowTracer Node Editor 2019.01 Build 71758 - denby1: denby\_201901@denby1

000039588 **INVALID** vov ./compute\_sizes ../../../../mkcdrom/cdrom pkg packages.tcl

Job Info Execution & Impact Annotations & Properties History Why Stdout +

Run Priority

Placement Policy fastest

Schedule Priority 8

Execution Priority 0

Xterm / Display

Runmode normal

X11 Display

Default X11 Display

Impact

	Valid	Other	Totals
Duration	0s	6m10s	6m10s
Files	0	13	13
Jobs	0	6	6
Unknown	0	0	0

Job Ownership (BETA)

☒ Allow LEADERs ownership

☐ Take Ownership

Sets and Origin of Job

Show Set (Containing This Job) in Set Viewer

Open Node Editor on Vovbuild Job That Created This Job (ID=1367)

Show Set of Jobs Created By Same Vovbuild Job in Set Viewer (ID=39365)

Figure 28:

The **Why** tab is available for both a Job Node and a File Node. This tab shows an explanation for the current state of the node.

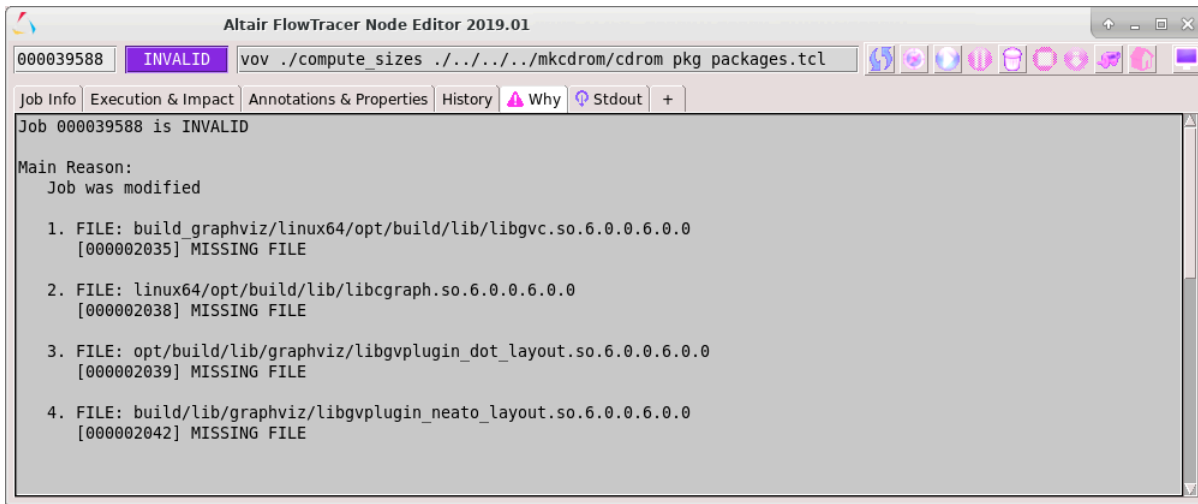


Figure 29:

## Horizontal Graph View of a Set

This view displays the nodes in the set as a graph showing their dependency relationship as arcs between the nodes. The graphical relationship has a horizontal perspective. The earliest dependencies in the graph are shown on the left of the graph, with latest dependencies shown on the right of the graph.

Open the Horizontal Graph view by clicking on the **Horizontal Graph** icon button  or by using the keyboard shortcut **shift-H**.

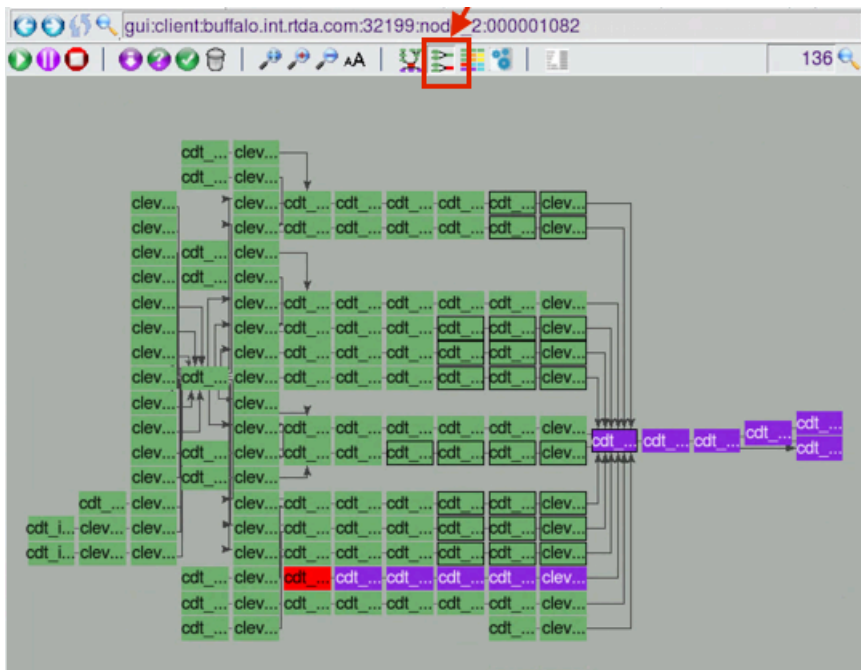


Figure 30:

The interaction with this view is the same as in the Vertical Graph view.

- Hover the mouse over a node; a pop-up dialog appears that describes that node.
- Left-clicking on a node selects it. A yellow border around the node indicates that it is selected.
- Right-clicking on a node displays a context menu. This menu provides options so you can perform selected operations on that node.
- Double-clicking the left button on a node opens the Node Editor in a separate window to show the properties of the current node.
- When the Node Editor dialog opens, it provides a set of tabbed pages. The default view for Job Nodes (rectangles) is the **Job Info** tab. The default view for File Nodes (circles) is the **File** tab.

## Grid View

This view displays the nodes of a set in a tight grid without regard to their location in the dependency graph. This is a graphical representation for the dependency graph in which arcs are not shown and the nodes are compactly arranged in a non-overlapping grid.

This display is effective when watching hundreds or thousands of jobs because it shows the changes of state in each node in a way that is easy to monitor. A node that fails has a distinct color that sticks out from the grid display. Progress through the nodes can be seen easily by setting the console preference in the **General** tab to enable "Move jobs to the bottom of the grid mode when they start". This shows the starting jobs accumulating along the bottom of the grid so that it is easy to monitor them starting, and noting their outcome.

You can switch to the Grid view by clicking the **Grid** icon  or use the keyboard shortcut **shift-Q**.

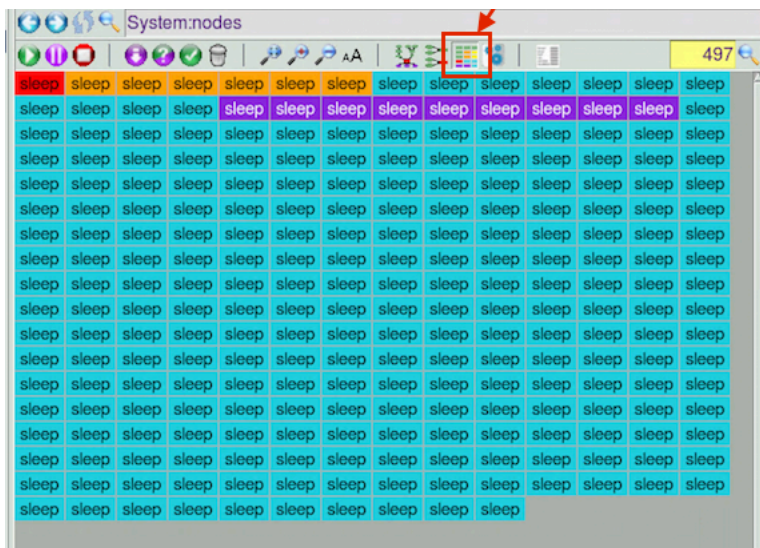


Figure 31:


While the appearance of the nodes in the window is different, the behavior is the same as the graph view:

- Mouse over a node; a descriptive label pops-up
- Right-clicking on a node displays a context menu

- Double-clicking on a node opens the Node Edit dialog.

Stat View

This view of the sets provides a tabular report on set attributes rather than showing the nodes that are members of the set.

Switch to the Stat view by clicking the **Stat** icon  or use the keyboard shortcut **shift-S**.

The Stat view offers a summary representation of the nodes in the selected set. This view is effective when you are working with millions of nodes and can only monitor a summary of them.

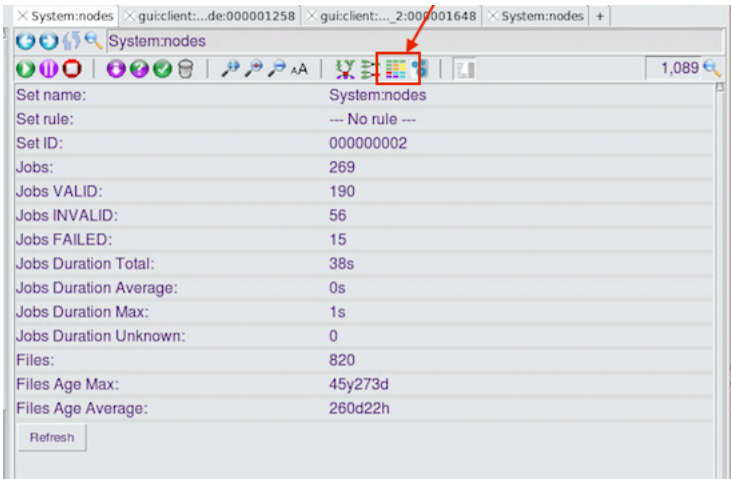


Figure 32:

Keyboard Shortcuts

To do this	Press
Scroll by 1 or 5 units in the direction of the arrow	Ctrl-Arrows
Node alone and expand: create a set containing only the current node, then expand to show the inputs and outputs of the node. This is a combination of the accelerator 'a' and the accelerator 'x'	A
Vertical Graph view	G
Detach selected nodes from the current set	D
Forget selected nodes	F
Horizontal Graph view	H

To do this	Press
Invalidate the selected nodes	I
Toggle Show/Hide Files	J
Grid view (sorry, not easy to remember)	Q
Stats view	S
Touch the selected nodes (toggle status twice between VALID and INVALID)	T
Expand the current set, showing all inputs and outputs of nodes in the current set.	X
Zoom out	Z
Node alone: create a set containing only the current node	a
Down-cone of the current set	d
Node alone and down-cone	Ctrl-d
Edit the current node	e
Fit the image to the size of the canvas	f
Show impact analysis for the current node	i
Select every node that has a similar status to any of the statuses in the current set of selected nodes	m
Show the navigation dialog for the current node	n
Pan the image so that the current location of the pointer is b at the center of the window)	p
Abort graph layout as soon as possible (interrupt weight-driven placement optimization) and show a partially optimized graph.	q
Retrace the selected nodes at normal priority. You must point to a node or a selection for this accelerator to have effect.	r
When hovering over a node in the node viewer, add it to the group of selected nodes.	s
Retrace the selected nodes at HIGH priority. You must point to a node or a selection for this accelerator to have effect.	R

To do this	Press
Up-cone of the current set	u
Node alone and up-cone	Ctrl-u
Expand the current node, showing its inputs and outputs	x
Zoom in. In graph and jobs views, if there is a selection, then the widget will zoom to best show the selected objects.	z
Select all nodes or blocks in the graph. See also Ctrl-i and Ctrl-c	Ctrl-a
Clear the selection (See also Ctrl-a and Ctrl-i)	Ctrl-c
Invert selection (See also Ctrl-a and Ctrl-c)	Ctrl-i

## Node Selection

To select nodes in the set viewer you have the following methods:

- Draw a rubber band around the nodes you want to select by clicking and dragging the left button.
- Hover over a node and click on the node to select.
- Hover over a node and type 's' to add it to the group of selected nodes.
- Type **m** to select every node with a similar status to any of the statuses in the current set of selected nodes.
- Press Ctrl-a to select all nodes in the viewer.
- Press Ctrl-i to invert the selection.
- Press Ctrl-c to clear the selection.
- Type **/** to activate the quick-search capability. Enter some other characters in the quick-search bar, then press Enter to select the nodes you have found.

## Weight Driven Placement

FlowTracer uses a weight-driven placement (WDP) algorithm to place nodes close to their dependencies and minimize overlapping edges. A higher weight is assigned to nodes with low fanout, which allows

nodes with single dependency relationships to stack perfectly. A lower weight is assigned to high fanout dependencies.

## Direction of Dependencies

Weight-driven layouts (graphs) can be placed in either a vertical or horizontal direction. Vertical graphs have primary inputs (files not generated in the flow) at the top, with dependencies going in the downward direction. The final outputs of the flow are displayed at the lowest position in the graph. The horizontal layouts flow left-to-right.

## Arrow Style

WDP supports 5 choices for dependency arrows:

Arrow Type	Description
<b>Auto</b>	Smaller graphs use curved arrows. Larger use straight lines or no lines. This is automatically determined based on the number of nodes and arrows in the graph.
<b>None</b>	Disables all arrows.
<b>Straight Lines</b>	Arrows are drawn center-to-center and may go behind other nodes.
<b>Curved (default)</b>	Curved arrows that do not overlap with nodes.
<b>Orthogonal</b>	Arrows with 90° angles that do not overlap with nodes.

## Controlling Size and Space of Nodes

In both vertical and horizontal layouts, WDP supports customizing the width and height of nodes and the spacing between the nodes. These values can be adjusted to produce the best looking graph for a given flow. These settings can be configured in the **Preferences** dialog on the **Graph** tab, and are independent between vertical and horizontal modes.

## Special Handling for Wide Graphs

It is common for a flow graph to include one or more dependency levels that contain significantly more nodes than the other levels. This proportion tends to produce graphs that appear very wide (in vertical mode) or very tall (in horizontal mode). There are 2 automated methods for altering wide rows to produce a graph with a better aspect ratio with nodes drawn larger and more meaningful.

Non-uniform node sizing is used to reduce the width of nodes (in vertical mode) or the height of nodes (in horizontal mode) for rows that have many nodes. The graph preferences include a minimum size setting that prevents reduced-size nodes from becoming too small to see.

The second method is to spread wide dependency levels onto multiple rows. In vertical mode, multiple rows can be used for a single dependency level. In horizontal mode, multiple columns can be used. The algorithm divides dependency levels until the aspect ratio of the result graph is close to the aspect ratio of the canvas window. The graph preferences include a minimum node count for rows to spread out. Only nodes with that many nodes or more are allowed to be spread onto multiple rows (vertical) or columns (horizontal).



## Quality vs Speed

Producing an optimal layout with the fewest possible arrow crossings is a complex problem. WDP provides three different choices to influence the trade-off of quality vs speed. The default method is to allow FlowTracer to choose the quality based on the size of the layout in terms of the number of nodes and arrows.

Quality Setting	Description
<b>Auto</b>	Layout quality is automatically determined based on the number of nodes and arrows in the graph. Smaller graphs are drawn in highest quality but very large graphs may not be optimized at all.
<b>Fast</b>	Layout quality is reduced for producing graphs as quickly as possible. This is best for large graphs to prevent delays in drawing.
<b>Balanced (default)</b>	Layout is partially optimized to reduce arrow crossings and edge lengths, but not completely.
<b>Good</b>	Graphs are placed with high quality (minimal arrow crossings and shortest arrows) even if it takes a long time to generate the layout.

## Graph Preferences

Preferences related to WDP layouts are found in the **Preferences** dialog, which can be accessed through the top level FlowTracer menu under **Console > Preferences** on the **Graph** tab, or by clicking the gear icon next to the buttons for vertical, horizontal, and grid view. The Control-Shift-G keyboard shortcut can also be used.

In the **Graph** tab, the choice for speed-vs-quality and arrow type can be modified. Below those, the Node Size and Spacing section allows modifying the size and spacing between nodes in vertical and horizontal mode. These settings are independent for each drawing direction.

The settings for allowing non-uniform node sizes and for spreading levels out to multiple rows is also controlled in the Node Size and Spacing settings. These settings are also independent for each drawing direction. Modifying this for vertical graphs has no effect on horizontal graphs.

In the **Graph** tab, there are 2 example graphs that show how changing the settings affects the layout. Some of the effects are not intuitive, such as how widening nodes tends to flatten the layout; the demos are drawn with Zoom Fit applied. Currently, non-uniform node sizes are supported in the demos but spreading rows is not supported.

Graph preferences can be saved to your home directory. Those settings will be applied to all future FlowTracer consoles opened by you. They can also be saved to the project; users who open a FlowTracer console on that project will load the settings that were saved.


The **Graph** tab includes buttons for reverting the changes that were made, loading the standard FlowTracer defaults, or forcing a Redraw of the current graph shown in the main FlowTracer Set Viewer (drawing window). It does not apply to sets that were floated using Float this Tab.

## Navigate the Flow Graph

Navigating the flow graph means moving from one node to another following the input/output dependencies.

You can follow the graph up and down the cone.

Activate the **Navigate** dialog in one of the following ways in the GUI:

- From the Node Editor, click the Compass icon 
- From the Set Viewer, move over a node, right-click to get a pop-up context menu and click **Connectivity > Navigate**.

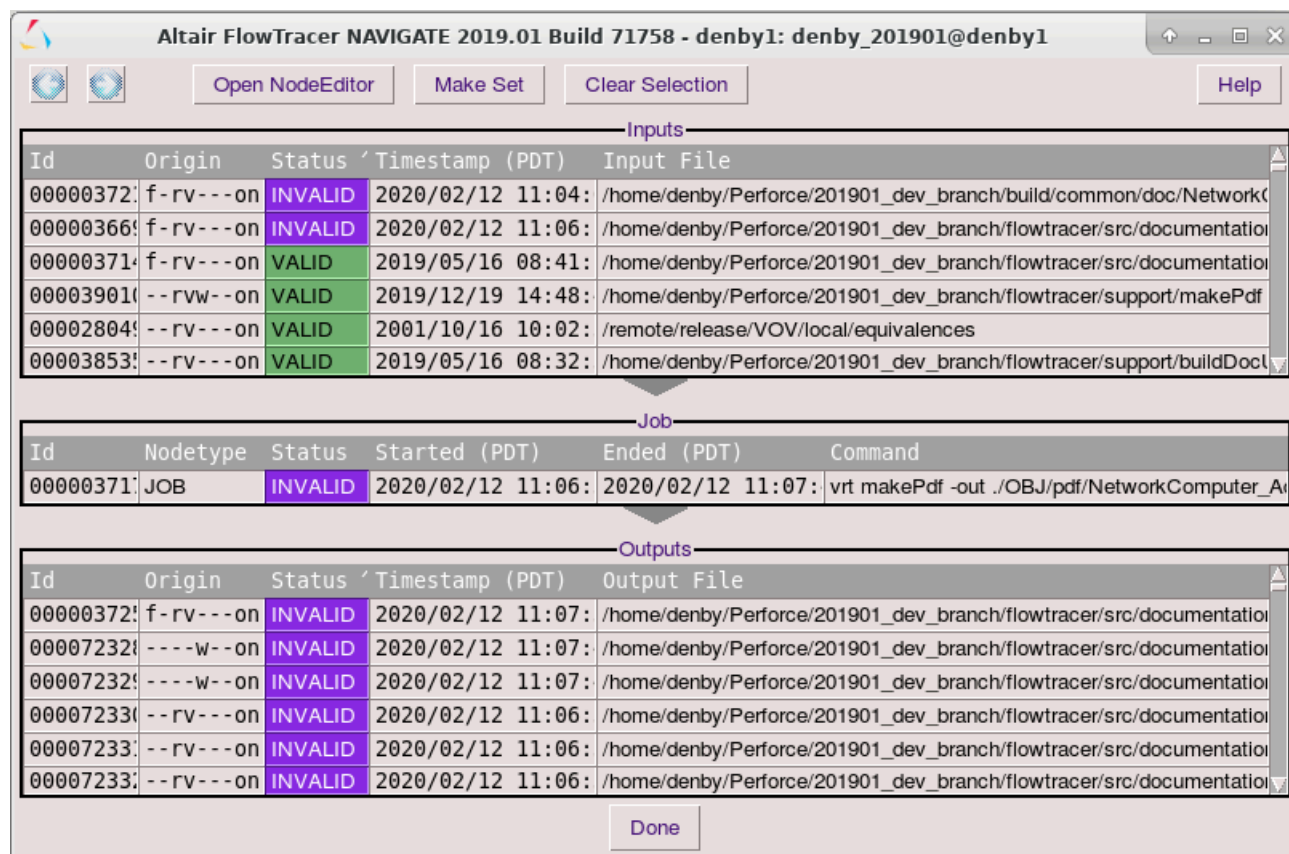


Figure 33:

A window opens that shows the node in focus within the center of the display, with a pane above showing the inputs for the node, and a pane below showing the outputs of the node.

Navigate up or down the cone by double-clicking on one of the inputs or outputs listed in the panes. This shifts the node in focus to the one you double-clicked on. It becomes the node in the center and its inputs and outputs are listed in the two panes.

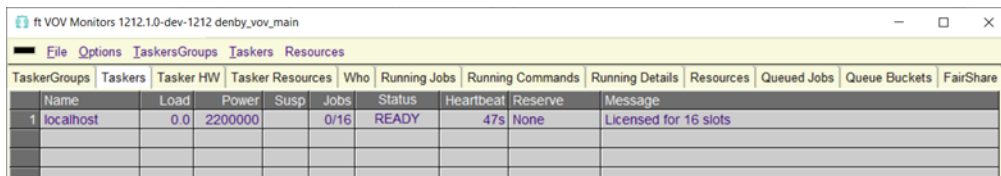
A right-click on an input or output pops up a context menu where you can choose to edit, navigate, or detach the node. Check the on-line help (<F1>) for more instructions.

At the command line, enter a CLI command to provide information to allow for similar navigation up and down the dependency cone.

```
% vsx [node-ID]
```

## Monitor Taskers

To monitor the activity of the taskers, use either `vsm` from the CLI, or click **Console > Vov Monitor** from the GUI.



TaskerGroups	Taskers	Tasker HW	Tasker Resources	Who	Running Jobs	Running Commands	Running Details	Resources	Queued Jobs	Queue Buckets	FairShare
	Name	Load	Power	Susp	Jobs	Status	Heartbeat	Reserve	Message		
1	localhost	0.0	2200000		0/16	READY	47s	None	Licensed for 16 slots		

Figure 34:

From the command line, you can use:

```
% vovtaskermgr show
1 04204992  bellevue 0.00      90909 0/1  READY  Unlim.
2 04205717      mars 0.00      39239 0/1  READY  Unlim.
3 04205719    saturn 0.05      10633 0/1  READY  Unlim.
4 04205720     moon 0.02      15003 0/1  READY  Unlim.
5 04205721   cayman 0.00      83333 0/1  READY  Unlim.
6 04205722    comet 0.07      58595 0/1  READY  Unlim.
7 04205725  cheetah 0.05     224089 0/1  READY  Unlim.
8 04205726   jupiter 0.03     13404 0/2  READY  Unlim.
```

From the browser interface, go to the Tasker page. For a large number of taskers, such as hundreds or more, you may find the Tasker page more compact and convenient.

### Run vovtasker monitor with CLI `vsm`

The command `vsm` is used to run the Tasker Monitor in its own window. This can also be called from the VovConsole or from any command line that has the right environment and is enabled for the project.

## `vsm`

Monitor activity in VOV.

```
vsm: Usage Message

DESCRIPTION:
    Monitor activity in VOV.

USAGE:
    vsm [options]

OPTIONS:
    -h                Show this help
```

```
-panel <panel>    Select initial panel (default 'running'),  
                  where panel is one of: running, resmaps, buckets,  
                  jobq, fairshare.  
-fontsize <size> Specify the font size. Default is 10.  
                  Legal range is 3 to 36.
```

OLD OPTIONS:

```
-t -r -q          Backwards compatibility
```

CONFIGURATION:

The starting panel can be configured in gui.tcl by means of  
the variable vovmonconfig(start,panel). Example:

```
# Fragment of $VOVDIR/local/gui.tcl  
set vovmonconfig(start,panel) "jobq"
```

EXAMPLES:

```
% vsm &  
% vsm -panel buckets &  
% vsm -panel taskers &
```

## Browser Interface

The URL to the FlowTracer web server is the entry point into the FlowTracer web application for managing the product. The URL is composed of a domain name and a port number and the path to the web application.

### URL to Browser Interface

The vovserver running in the FlowTracer project also provides a web server as part of its feature set. Interacting with this web server is a powerful set of web application components that interact with a browser. They provide the full set of tools to manage FlowTracer.

The URL is:

```
http://host:port/project
```

You can discover the URL to access the browser interface for your project by asking FlowTracer to tell it to you using any of these CLI commands: `vovbrowser`, `vsi`, or `vovproject info`.

### Use vovbrowser to Discover the URL

You can use the command `vovbrowser` to get the URL to enter into your browser to reach the server. Without any option, `vovbrowser` prints the URL value for the current project: For example:

```
% vovbrowser          # Use current project parameters.  
http://alpaca:6276/project
```



**Note:** If your FlowTracer's `setup.tcl` file is configured with `VOV_HOST_HTTP_NAME`, it will be shown by `vovbrowser`, instead of the value of `VOV_HOST_NAME`.

### Use vsi to Discover the URL

You can also use the "get-information" command `vsi` to get the URL to enter into your browser. The URL is one of the many project attributes that is reported.



**Tip:** This command should be the first thing to check when investigating vovserver performance issues.

Bottom right section shows URL to access the server:

```
% vsi  
someproj@bison          | 09/04/2015 10:15:40  
-----  
Files: 39027             | Sets: 367  
Jobs: 19776              | Retraces: 78  
valid: 13533             |  
invalid: 6220            |  
running: 3               |  
retracing: 4             |  
sleeping: 10             |  
failed: 6                |  
-----  
Taskers: 14              | QueuedJobs: 3464
```

Slots: 16		Buckets: 57
ready: 10		Duration: 9h28m
full: 3		
busy: 1		
-----		
Resources: 112		URL: http://bison:6244
Exhausted: 1		Saved: 1h44m
Priority:normal		Size: 33.00MB
		TimeTol: 3s
-----		
Recent jobs for user john		
04485265	VALID	vw2 CC -mt -o OBJ/sun7-gmt/trace.o -c -g -...
04485268	RUNNING	vw CC +eh -o OBJ/hpux-O/securitysrv.o -c -...
04485270	VALID	vw2 CC -mt -o OBJ/sun7-gmt/tracecommclient...
04485273	RUNNING	vrt g++ -mcpu=common -Wall -o OBJ/linux-g/t...
04485276	RUNNING	vw cl -nologo -FoOBJ/win64-O/fairshare.obj...
04485277	VALID	vw2 CC -mt -o OBJ/sun7-gmt/tracecomm2.o -c...

The program `vsi` is equivalent to this command:

```
% vovproject info
```

## Project Home Page

When you enter the URL to access the browser interface you will start the web application to manage the system. The first page of the web application is the **Project Home** page. It offers a top-level menu of features you can navigate.

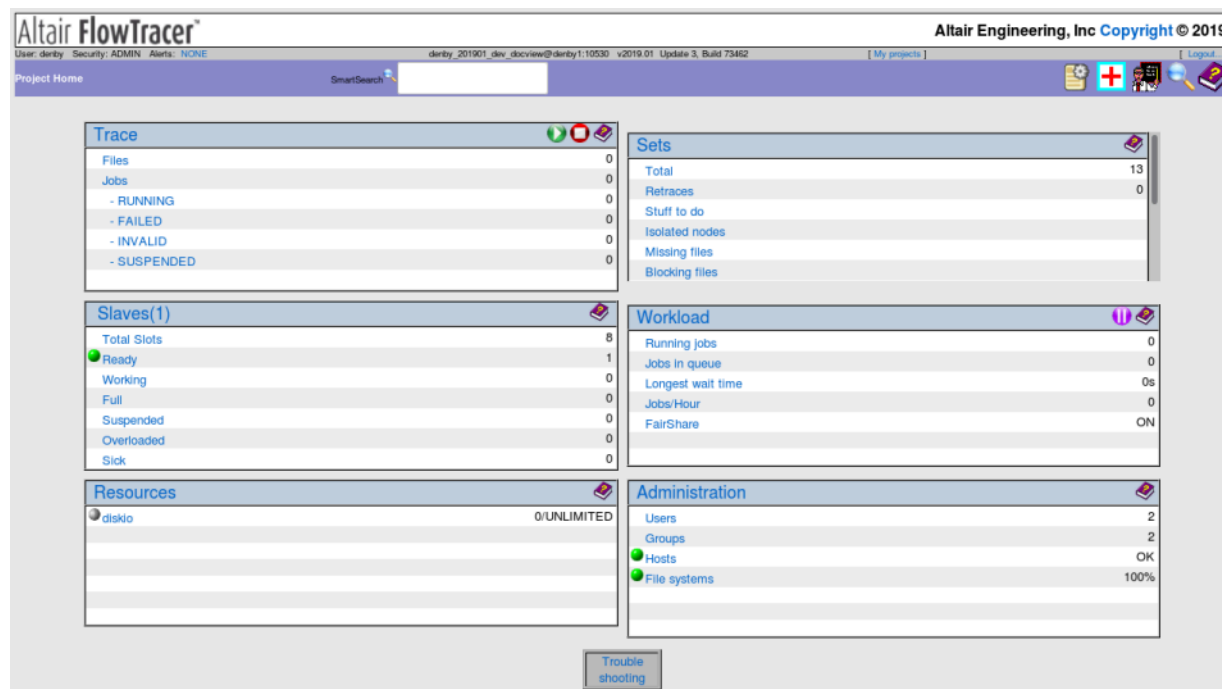


Figure 35: Project Home Page

The six major areas of managing the system are accessible by clicking on the names in the headers of each of the six tables in the display.

Trace

Drill down to see information about the job and file nodes registered with the project. The breadcrumb navigation shows that this view comes from looking at the members of the all-inclusive set, **System: nodes**. This is the system-defined set that holds all nodes in the graph.

Each node ID in the list provides a link to navigate to the **Node Detail** page.

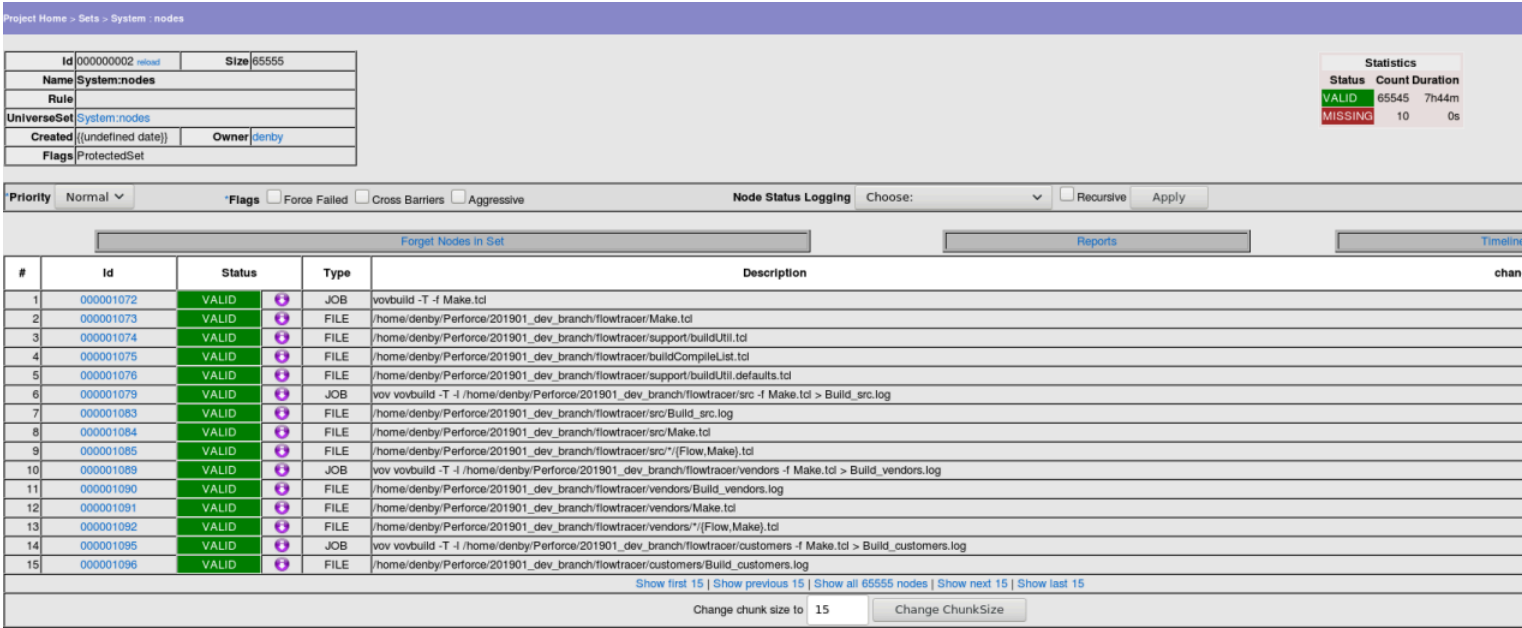


Figure 36:

Sets

Sets is a way of grouping nodes in the dependency graph by gathering together nodes that match the selection rules for the set. That group of nodes can be accessed by viewing the members of that set. Each set name in the list provides a link to navigate to the **Set Detail** page.











		Status	Size	Duration	Actions
	[ @@@ ]				
	[ All ]				
	[ CDROM ]				
	[ Class ]				
	[ FlowTracer ]				
	[ Predefined ]				
	[ System ]				
	[ src ]				
	[ test ]				
	[ unit_testing ]				
	[ vendors ]				
1	CDROM	434	434	0s	    
2	vendors	44	44	1m03s	    

Figure 37:

Taskers

This page displays a table listing the taskers with information about the current state of each tasker host. Each row in the table provides links from the group ID and the server host name to see details about each tasker host. The tabs across the top of the **System Monitor** page lead to other panels showing details about those elements.

Project Home > Slaves

SmartSearch

Slaves

Slave Resources

Running Jobs

Retraces

Job Queue

Resources

Fair Share

<input type="checkbox"/>	#	License	Group	Name	Power	Load	Jobs ?	Status	Heartbeat	Message
<input type="checkbox"/>	1	XNPR	g1	dispatcher	1856203	0.48	0/8	READY	23s	
<input type="checkbox"/>	2	XNPR	g1	worker1	1848872	0.48	0/8	READY	23s	
<input type="checkbox"/>	3	XNPR	g1	worker2	1855448	0.48	0/8	READY	23s	
<input type="checkbox"/>	4	XNPR	g1	worker3	1855204	0.48	0/8	READY	23s	
Showing 4 of 4 slaves					Total		0/32			

Start all slaves

Stop selected slaves

Kill jobs on selected slaves

Refresh environments

Plot Slave Load

Show Compact Slave Monitor

Show Slave Lists

Select slaves based on their compatibility with the following request for hardware resources:

Figure 38:

Workload

This table is empty when the system is idle and up-to-date. When the system is running, each row in the table reports on a Job Queue Bucket, which is a grouping of jobs waiting for the same collection



of resources. There is one bucket for each unique collection of resources. Jobs in the queue that are waiting for that collection of resources are put into that bucket, along with any other jobs waiting for the same collection of resources.

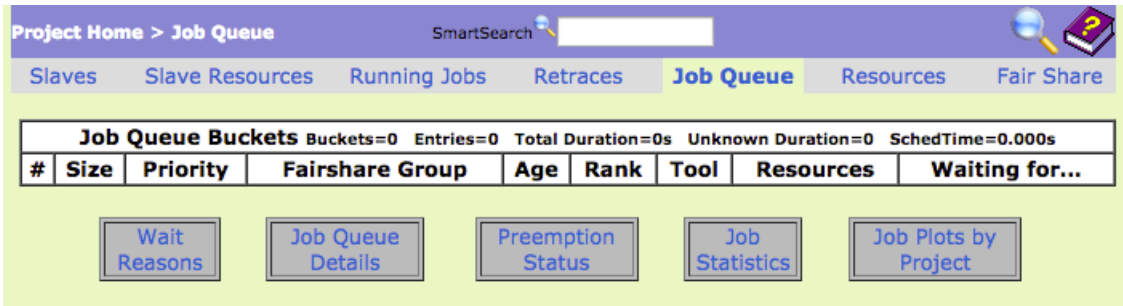


Figure 39:

Resources

Each row in the table provides links to see details about each resource. The tabs across the top of the **System Monitor** page lead to other panels showing details about those elements.

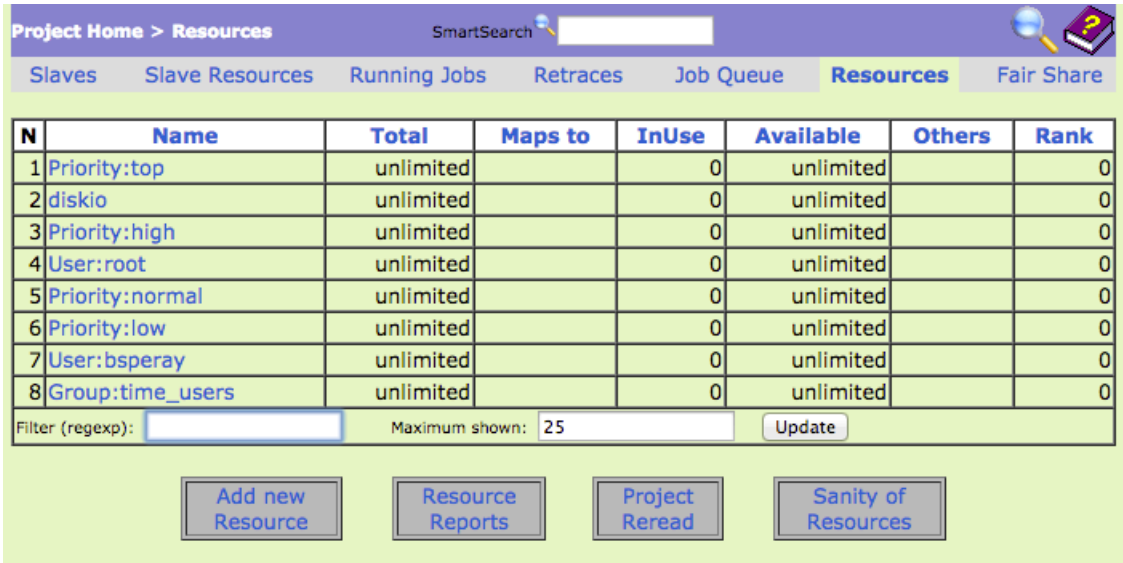


Figure 40:

Administration

This page is aimed at the Administrator User. You can learn more about the Administrative functions available in this section of the application in the Administrator's manual.

Project Home > Admin
SmartSearch

**Actions**

- Save
- Reread
- Sanity
- Shut Down

**Administration**

- Alerts
- License
- System Recovery
- Database
- Daemons
- Environments
- Periodic Jobs
- Users
- Who
- Groups
- Web-based setup
- Live Recorder
- Show Patches

**Network**

- Equivalences
- Equiv. Caches
- File Systems
- Hosts
- Processes

**Internal Data Structures**

- Server Environment
- Server Config Params
- Vov Protocol Statistics
- Current Clients
- Project Registry

General Information	
Project Name	hourglass
Product Name	ft
Server Type	primary
Host	mac23
Port	15918
Web Port	6416
Read-Only Port	0 OFF
Working Directory	\${HOME}/vov
Server Version	MacOS/2015.09 Build 48713 built:Oct 13 2015 14:34:14
Server Size	9,965,568 bytes (estimated)
Server Delay	8s
*Server Capacity	256
Automatic Shutdown	13d22h
Project Start	Fri Aug 21 16:16:08 2015
Last Start	Tue Oct 13 17:05:45 2015
Last Save	Fri Oct 16 17:16:02 2015
Log File	/Users/bsperay/vov/hourglass.swd/logs/server.2015.10.16
Log Size	544 bytes
*Automatic Rerun Threshold	0s (disabled)
Change Auto-rerun Threshold	0s   1s   2s   3s   4s   10s   20s   40s   60s

Figure 41:

### Configuration Parameters

The FlowTracer system uses Configuration Parameters to manage actions and behavior and to constrain various resources. The page showing the set of Configuration Parameters is accessed by clicking **Server Config Params** from the lower left side on the main page



<div> <div>User: bsperay Security: ADMIN Alerts: hourglass@mac23 v2015.09 Build 48791</div> <div>[ My Projects ] [ Logout... ]</div> </div>							
<div> <div>Project Home &gt; Admin &gt; Server Configuration Parameters</div> <div>SmartSearch </div> </div>							
	Parameter 	Value	Config File	Type	Default	Range	Desc
1	alerts.max	50	policy.tcl	integer	50	[5--1,000]	Maximum number of alerts that can be archived.
2	allowcoredump	1	policy.tcl	boolean	0	0,1	If nonzero, allow core events where allowed.
3	allowUidForSecurityFile	0	policy.tcl	integer	0	[0--2,147,483,648]	Additional file.
4	autoForgetFailed	2d00h	policy.tcl	timespec	2 days	[5m,infinity]	If the alert will be forgotten after the specified time.
5	autoForgetOthers	2d00h	policy.tcl	timespec	2 days	[5m,infinity]	If the alert will be forgotten after the specified time.
6	autoForgetRemoveLogs	0	policy.tcl	boolean	0	0/1	Try to remove logs after auto-forgetting.
7	autoForgetValid	1h00m	policy.tcl	timespec	1 hour	[5m,infinity]	If the alert will be forgotten after the specified time.
							The period of time that the system will...

Figure 42:

### Adding More Clients

As a user you may reach the maximum number of clients allowed per user with FlowTracer. This maximum number is controlled by the `maxNormalClients` configuration parameter, which is set to a default value of 400. This configuration control is used to protect against a Denial of Service attack.

You may need to increase this maximum by navigating to the Server Configuration Parameters list and scrolling down to that parameter and increasing the value.

67	maxJobArray	10,000	policy.tcl	integer	10,000	[10,100000]	This parameter defines the maximum of a job array.
68	maxLevel	1,023	policy.tcl	integer	1,023	[3,1023]	Maximum number of levels in the dependency graph. Dependency declarations that cause the levels in the graph to exceed this value are rejected.
69	maxNormalClients	400	policy.tcl	integer	400		Maximum number of normal clients per user. This is a protection against denial-of-service attacks by a single user.
70	maxNotifyBufferSize	400,000	policy.tcl	integer	1,000,000	[100k,40M]	In bytes, this parameter controls the size of the buffers used internally to notify asynchronous clients. If you see "overflow" events and are annoyed by them, you can increase the size of the buffers.
71	maxNotifyClients	40	policy.tcl	integer	40		Maximum number of notify clients per user. This is a protection against denial-of-service attacks by a single user. The 'notify' clients are those that tap into the event stream. They are used for example in waiting for jobs to finish and to update the GUI.
72	maxPathLength	1,024	policy.tcl	integer	1,024	[128,16000]	Maximum length of names of files.

Figure 43:

## Customize the Web User Interface

Using the `webgui.json` file, you can customize the FlowTracer web user interface to your own preference. The following instructions explain how to use the file to make some changes.

Copy the generic example file `webgui.json` from the `/etc/config/ftwebui` location into your SWD root folder. This immediately effects the web UI.

### Edit the Predefined Custom Actions

There are 4 reserved custom actions that can be triggered when executed from the FlowTracer Web UI.

1. Invalidate
2. Force Validate
3. Waive Exit Code
4. Skip

The Web UI context menu has already implemented these actions, but each of them has a predefined behavior that can be extended by adding a custom action with the reserved keys `"invalidate"`, `"forcevalidate"`, `"waiveexitcode"` and `"skip"`.

**Tip:** These actions work similarly as described in the section "Customization callback for `vovconsole`" in the `gui.tcl` file. That section applies to `vovconsole` and can be defined for the FlowTracer Web UI from here.

To add a new custom action, add the name as key and fill in the 4 required fields (`command`, `env`, `logfile` and `rundir`).

```
"customactions": {
```

```
"forcevalidate": {
  "command": "vw /usr/bin/touch /tmp/my_custom_action",
  "env": "BASE",
  "logfile": "/tmp/myCustomAction.log",
  "rundir": "."
},
"waiveexitcode": {
  "command": "",
  "env": "",
  "logfile": "",
  "rundir": ""
},
"skip": {
  "command": "",
  "env": "",
  "logfile": "",
  "rundir": ""
},
"invalidate": {
  "command": "",
  "env": "",
  "logfile": "",
  "rundir": ""
}
}
```

## Add New Custom Actions

Extra custom actions can be defined by filling in the "custommenus" array.

Each custom menu entry defines a new action, as well as how and where the FlowTracer Web UI should display it.

Each entry has 6 fields:

1. command
2. env
3. logfile
4. rundir
5. name
6. objectscope

The first 4 fields (command, env, logfile and rundir) work the same as in the Custom Action definition. The "name" field defines the label displayed in the Web UI and the "objectscope" field defines where the new menu item will appear ("sets | nodes | files").

```
"custommenus": [
  {
    "command": "",
    "env": "",
    "logfile": "",
    "rundir": "",
    "name": "label 1",
    "objectscope": "sets | nodes | files" // will appear in all objects
  },
  {
    "command": "",
```

```
    "env": "",
    "logfile": "",
    "rundir": "",
    "name": "label 2",
    "objectscope": "nodes" // will appear only in nodes (job or files)
  },
  {
    "command": "",
    "env": "",
    "logfile": "",
    "rundir": "",
    "name": "label 3",
    "objectscope": "sets" // will appear only in sets
  },
]
```

## Command Line Interface

The Altair Accelerator products have a rich set of programs that can be run on the shell command line.

Each program performs a specific task and is controlled by a set of command line parameters. The programs provide a usage command when run with a -h parameter as a quick way to understand what each does and what parameters it takes.

This section describes the programs that a user will find useful. There are other programs described in the Developer and Admin manuals that are useful for their situations.

### Common Commands

vovproejct	Create, start, enable, stop a project.
vovconsole	Start the Graphical User Interface (GUI).
vovcheck	Run many checks on the setup of VOV.
vls	List the status of the files with respect to the dependency graph.
vsc	Show the consequences of changing one node.
vsh	Report the history of a file or a job.
vsi	Report the basic Information about the graph.
vsm	Monitor taskers, jobs, resources.
vsr	Issue a retrace request.
vst	Show tools executed in the current directory.
vsx	Show inputs and outputs of nodes.
vsy	Show why a node is not VALID.
vsz	Zap away bad nodes.

### Main User Commands

vovblast	Simultaneously remove and forget files
vovbrowser	Report the URL to start the browser interface, including the right port number for a project.
vovbuild	Build complex flows.

vovcleandir	Eliminate old log files that are no longer part of the flow.
vovfind	Find file nodes in the graph.
vovfire	Execute a job "here and now".
vovforget	Forget objects in the in-memory database: nodes, sets, users, etc.
vovkill	Kill a process and all of its children.
vovjobqueue	Show all jobs in the queue.
vovrename	Rename nodes in the graph.
vovset	Create, edit, forget sets.
vovsh	The main client, with extended Tcl interpreter.
vovshow	Show things: users, buckets, nodes, sets, etc.
vovstop	Stop things: jobs, projects, retraces, ...
vovtouch	Toggle status of nodes.
vovversion	Show the version of the current installation.

## Other Commands

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



## List Files in Directory - Showing Project Status

### The CLI Program `vls` - Vov List Files in Project

The command line program `vls` lists the files in the current directory with respect to their role in the current project. All files in the directory are listed. Files from the project are shown with information about their state and role within the project. Other files are labeled to indicate they are not used within the project.

With no options, `vls` shows the [status](#) of the file, its role in the project's dependency graph and its name. The label "UNKNOWN" is used to indicate files that have no role in the project.

The project role is indicated by a one letter code:

- **i** - primary input
- **o** - primary output
- **u** - used in the flow but not primary input or output

### List Files in Current Directory

```
% vls
VALID      i   aa
VALID      o   bb
            UNKNOWN bbb
VALID      u   cmd_vls_help.txt
VALID      u   cmd_vovassist.txt
VALID      u   cmd_vovcapsule.txt
```

### List Files using LONG Format

Options `-long` and `-LONG` give more information for files in the project.

The option `-long` or `-l` reports on project files in a long format, showing their level, their VovId, their project role, their status and their name.

The option `-LONG` or `-L` reports on project files in an even longer format, showing their level, their VovId, the number of their inputs and outputs, their status and their name.

```
% vls -L
1 03182557 ( 0, 1) VALID      Flow.tcl
3 03778456 ( 0, 24) VALID     buffer.cc
            UNKNOWN buffer.cc.~1.6.~
1 03778453 ( 0, 82) VALID     buffer.hh
            UNKNOWN buffer.hh.~1.2.~
3 00003662 ( 0, 62) VALID     bufperf.hh
```

### List Specific Files and Files in a Specific Directory

The command line program `vls` can also be called to list information about specific files and directories of the project, referencing them by name.

The arguments following the options are interpreted as:

- files
- directories

You can combine any number of files and directories and you can use file name wild-card characters of '?' and '\*'.

```
% vls aa bb ../dir1 *.txt
VALID      i   aa
VALID      o   bb
VALID      i   ../dir1/Flow.tcl
VALID      u   ../dir1/buffer.cc
UNKNOWN    ..../dir1/buffer.cc~1.6.~
VALID      u   ../dir1/buffer.hh
UNKNOWN    ..../dir1/bufferf.hh!1.2.~
VALID      u   ../dir1/bufferf.hh
VALID      u   cmd_vls_help.txt
VALID      u   cmd_vovassist.txt
VALID      u   cmd_vovcapsule.txt
```

## vls

Display information about the files. By default, all files in the current working directory are listed.

vls: Usage Message

DESCRIPTION:

Display information about the files. By default,  
all files in the current working directory are listed.

USAGE:

vls [OPTIONS] [file or dir] ...

OPTIONS:

-help -- Print this message  
-long -- Give info in long format  
-LONG -- Give info in even longer format  
-Output -- Specify output format

Options can be truncated to 1 or 2 characters.

FORMATS:

default: @STATUS:10@ @ZIPPED@ @IUO@ @NAME@  
long : @LEVEL:3@ @ID@ @IUO@ @STATUS:10@ @ZIPPED@ @NAME@  
LONG : @LEVEL:3@ @ID@ (@INPUTS:4@,@OUTPUTS:4@)  
 @STATUS:10@ @ZIPPED@ @NAME@

EXAMPLES:

```
% vls
% vls -l
% vls foo*
% vls dir1 ../dir2
% vls -O "@ID@ @STATUS@ @TAIL@"
```

## List Jobs from Project Having a Given Directory

### The CLI Program vst - Vov Show Jobs (aka Show Tools)

The command line program `vst` lists the jobs from the project that were executed within the indicated working directory, with a default of also having the current environment.

A job in a project might be referred to as *Running a Tool*, so the name for this program was created to be an acronym for "Vov Show Tool."

```
vst: Usage Message

DESCRIPTION:
  Show the jobs that have been executed
  in one or more directories.  By default:
  - look at the current directory.
  - show only the jobs with same environment as the current one.

USAGE:
  vst [OPTIONS] [dir1] ...

OPTIONS:
  -help           -- Show this help
  -quiet          -- Be quiet, no explanations
  -fast           -- (Obsolete option: Maps to -all)
  -all            -- Show all jobs, regardless of the environment.
                   Also show system jobs and subjobs.
  -long           -- Show more information (LEVEL and ENV)
  -LONG           -- Show even more information
  -Output 'format' -- Choose output format
  -systemjobs     -- Show also the system jobs
  -subjobs        -- Show also subjobs (components of a bigger job)
  -regex <expr>  -- Filter lines with a regular expression
  -sort           -- Sort lines

Options can be truncated to 1 or 2 characters.
Option -regex can also be abbreviated to -rx.

EXAMPLES:
% vst
% vst -all
% vst -a
% vst -long ..
% vst -O '@LEVEL@ @COMMAND@'
% vst -re 'ab[cC]+'

```

### Show Jobs for Current Directory and Environment (BASE)

By default, the jobs in the current directory and with the current [environment](#) are listed:

```
% vst
vst: rule    is 'ISJOB==1 ENV==BASE CWD==${SEV_PROJECT_DIR}/common/src/doc_html'
vst: format is '@ID@ @STATUS:10@ @COMMAND@'
00387448 VALID      vov ./build.tcl index.tcl
00392249 VALID      vov ./makeindex.tcl
00425481 VALID      clevercopy work/index.html ../../doc/html/index.html

```

## Show Jobs for Current Directory and ANY Environment

Use option -all or -a to list all jobs regardless of the environment.

You can change the output format using either the option -l, -L or -O.

```
% vst -a
vst: rule    is 'ISJOB==1  CWD==${SEV_PROJECT_DIR}/common/src/doc_html'
vst: format is '@LEVEL:3@ @ID@ @STATUS:10@ @ENV:8@ @COMMAND@'
  2 00587448 VALID      GNU      vw gcc -c -g strtr.cc
  2 00587448 VALID      GNU      vw gcc -c -g amos.cc
  2 00387448 VALID      BASE     vov ./build.tcl index.tcl
  4 00392249 VALID      BASE     vov ./makeindex.tcl
  6 00425481 VALID      BASE     clevercopy work/index.html ../../doc/html/index.html
```

## Start Run - Run Jobs Depending on Changed Inputs

### The CLI Program vsr - Vov Start Run

The command line program `vst` triggers the FlowTracer system to generate a runtime trace based on the dependency graph in order to schedule and run jobs that depend on the changed inputs. The system will continue this until no more jobs need to run because of changed inputs.

The name of this program was created to be the acronym for "Vov Start Retrace".

This command is used when input changes are ready to apply, and you want all the dependency processing to start running. This is similar to requesting a recalc in a spreadsheet when all the changes have been made.

### vst

Retrace a list of files and directories. Use option -set to retrace sets of nodes.

```
vst: Usage Message
DESCRIPTION:
  Retrace a list of files and directories.
  With the option '-set' you can retrace sets of nodes.

USAGE:
  % vst -help
  % vst -set <setName>
  % vst [options] <id_or_file_or_directory> ...

OPTIONS: All options can be abbreviated to one letter,
         except -safe, -fast, -aggressive, and -select.

NORMAL FLAGS:
  -help          This message.
  -v             Increase verbosity.
  -all           Retrace the entire design (abbreviation -a).
  -force        Force retracing of failed nodes too.
                Abbreviation: -f
  -hier         If target is a set with subsets, retrace all subsets.
```

```
-kick          Kick a file: update all files that depend on it.
-priority sp[.xp]
                Specify the scheduling priority (sp) and optionally
                also the execution priority (xp).
                Both sp and xp can be one of:
                LOW, NORMAL, HIGH, TOP or an integer in [1-15].
                (Explanation: TOP=15  HIGH=8  NORMAL=4  LOW=1)
-resources <r> Additional resources to be used for the jobs to be
                retraced.  Abbreviation: -r
-select <rule>  Retrace nodes that match the given selection rule.
-set <setName>  Retrace all nodes in given set.
-recompute     If used with -set, forces recomputation of named set.
-timeout <timeSpec>
                Wait for events for specified time, then exit if
                no event arrives.
-nowait        Exit immediately after issuing retracing request.
                This is the same as "-timeout 0"
```

ADVANCED FLAGS:

```
-safe          All dependencies are computed again (default).
-fast         Retraced without computing the dependencies again.
-aggressive    Like safe, but retrace in parallel across barriers.
-crossbarriers Schedule jobs across barriers
                Abbreviation: -cb
-skipcheck     Do not check the upcone of the target.
-checkall      Check all files in the upcone of the target.
-nocheck       Do not check files in directories to retrace.
                Implies -skipcheck.
-unblock       Check blocking files before doing retrace.
-retry <n>     Repeat retracing <n> times. The default is 1.
-converge      Retry until the flow is complete or until the
                failed/invalid job count is unchanged after 3 retrace
                loops. If a retry number is specified, converge will
                limit the number of retrace loops to that number.
-statusreps <n>
                Override the default of 3 for the maximum number of
                repeated failed/invalid job counts for the converge
                option.
```

EXAMPLES:

```
% vsr -help
% vsr
% vsr -priority high.high fileABC
% vsr . -crossbarriers
% vsr . -cb
% vsr ../a ../b ../c/file
% vsr -nowait -all
% vsr -timeout 3h
% vsr -retry 4 -timeout 100
% vsr -set 'Stuff to do'
% vsr -recompute -set 'Stuff to do'
% vsr -select 'isjob tool==gzip'
```

## Examples

You can request a run for a given target file with:

```
% vsr fileName_or_VovId
```

You can run all jobs in a directory with:

```
% vsr directoryName
```

You can target a run to be a group of nodes that make up an entire set with:

```
% vsr -set setName
```

As a special case, you can run everything in the whole design. Use the system defined private set System:nodes with:

```
% vsr -all
```

Set the Priority with the option -p:

```
% vsr -p high directoryName
```

If you change a file and would like to retrace starting with that changed file, running jobs that are affected by the change, use:

```
% vsr -kick name_or_VovId_of_changed_file
```

## Show Information About Current Project

The command line program `vsi` reports basic statistics about the current project.

The name of this program was created to be the acronym for "Vov Show Info."

### **vsi**

Print basic information about a project. Also useful to test whether a vovserver is running or not.

```
vsi: Usage Message

DESCRIPTION:
    Print basic information about a project.
    Also useful to test whether a VOVserver
    is running or not.

USAGE:
    % vsi [options]

OPTIONS:
    -help                -- This message.
    -width <N>           -- Size of command line strings (default 50).
    -j                   -- Show the most recent 6 jobs of mine.

EXAMPLES:
    % vsi
    % vsi -j
```

```
% vsi -width 100
% vsi -w 100
```

## Examples

For project that builds a product having a large source code repository:

```
% vsi
someproj@bison | 09/04/2015 10:15:40
-----
Files: 39027 | Sets: 367
Jobs: 19776 | Retraces: 78
valid: 13533 |
invalid: 6220 |
running: 3 |
retracing: 4 |
sleeping: 10 |
failed: 6 |
-----
Taskers: 14 | QueuedJobs: 3464
Slots: 16 | Buckets: 57
ready: 10 | Duration: 9h28m
full: 3 |
busy: 1 |
-----
Resources: 112 | URL: http://bison:6244
Exhausted: 1 | Saved: 1h44m
Priority:normal | Size: 33.00MB
| TimeTol: 3s
-----
Recent jobs for user john
04485265 VALID vw2 CC -mt -o OBJ/sun7-gmt/trace.o -c -g -...
04485268 RUNNING vw CC +eh -o OBJ/hpux-O/securitysrv.o -c -...
04485270 VALID vw2 CC -mt -o OBJ/sun7-gmt/tracecommclient...
04485273 RUNNING vrt g++ -mcpu=common -Wall -o OBJ/linux-g/t...
04485276 RUNNING vw cl -nologo -FoOBJ/win64-O/fairshare.obj...
04485277 VALID vw2 CC -mt -o OBJ/sun7-gmt/tracecomm2.o -c...
```

This is equivalent to:

```
% vovproject info
```

## Show the History of a File or Job Within the Project

The command line program `vsh` reports on the history of a file or job with respect to the current project. It can be run to report on more than one file or job by listing multiple identifiers on the command line. You can use command line options to control the scope of the report, such as the number of levels of history to show, or if notes/annotations for the item should be shown.

You can pass in a name of a file or the VovId of a node as the item to report on.

When this is called to show the history of a file, you can see the jobs it was involved in either up or down the dependency graph by using the `-backward` or `-forward` options.

When this is called to show the history of a job by passing in the VovId of the job node, you can see the files it was involved in either up or down the dependency graph by using the -backward or -forward options.

The name of this program was created to be the acronym for "Vov Show History."

## vsh

Show history of files in the dependency graph.

```
vsh: Usage Message

DESCRIPTION:
    Show history of files in the dependency graph.

USAGE:
    % vsh [options] <id_or_file> ...

OPTIONS:
    -help                -- Show this usage message.
    -level <n>           -- Show up to <n> levels of history.
    -backward            -- (Default) look upstream
    -forward             -- Look downstream (dependent nodes).
    -annotations         -- Show annotations.
    -files               -- Show files.
    -jobs                -- Show jobs.
    -nodes               -- Show all nodes.
    -notes               -- Same as -annotations.

EXAMPLES:
    % vsh file_aa
    % vsh -level 10 -forward -files 00022345
```

## Examples

```
% vsh bb.txt
${TOP}/bb.txt
VALID Mon Oct 20 14:37:10 2015

00000364 VALID vw cp aa.txt bb.txt

% vsh 000392
# -- Environment: DEFAULT
cd ${TOP}
  vw cp bb.txt cc.txt
# -- Started      : Tue Oct 13 07:21:39 2015
# -- Ended       : Tue Oct 13 07:21:40 2015
# -- Id          : 00000392
# -- Pid        : 205
# -- User       : johnX
# -- Host       : lucca
# -- Duration(expected): 1(1)
# -- Resources  :
# -- Priority    : UNDEF
# -- Exit status: 0   Allowed: 0
```



```
00000392 VALID vw cp bb.txt cc9.txt
00000364 VALID vw cp aa.txt bb.txt
```

## Impact of Changing a File or Node

### The CLI Program vsc - Vov Show Consequences

The command line program `vsc` generates an "Impact Analysis" report to estimate the consequences of changing a particular file or node.

The name of this program was created to be the acronym for "Vov Show Consequences."

### Impact of a Node Changing

When downstream jobs depend on an input file that changes, the downstream jobs must be run again. As they run, they can cause files to change which in turn trigger a need to run other jobs, and so on. All this downstream effect based on the dependency graph is known as the [down-cone](#) of a node.

The impact of a node is the total expected duration of the jobs that are rerun in its down-cone.

FlowTracer uses impact analysis to schedule jobs in the critical path ahead of the other jobs.

For efficiency reasons, FlowTracer uses an approximate calculation of the impact. In particular, for the purpose of job scheduling, all jobs with duration less than 5 seconds, are categorized as a "large" impact, thus saving the computation of a full and accurate impact.

## VSC

This script performs "Impact Analysis" with VOV.

### Description

vsc: Usage Message

#### DESCRIPTION:

This script performs "Impact Analysis" with VOV. It shows the possible consequences of changing something in the design. In general, the consequences are that all dependent jobs must be executed again.

The report is a worst-case analysis, in the sense that it is assumed that changes will propagate through all barriers. (For an explanation of barriers, see `clevercopy`).

#### USAGE:

```
% vsc [options] <node_descriptor> ...
```

where the node descriptor is either the name of a file or the VovId of a node in the dependency graph.

#### OPTIONS:

```
-help          -- This message.
```

EXAMPLES:

```
% vsc a
% vsc 00023456 00023457
```

## Why a Node Has its Current Status

The command line program `vsy` reports on why a given node has its current status.

The name of this program was created to be the textual abbreviation for "Vov Status Why."

### **vsy**

Analyze job status reasons.

`vsy`: Usage Message

DESCRIPTION:

Analyze job status reasons.

USAGE:

```
% vsy [-failed | -invalid | -help] <jobid | ! | set | dir | file> ...
```

COMMON OPTIONS:

```
-help                -- Print this message
-maxreasons <number> -- Occasionally, there can be more than
                        one reason for nodes to be INVALID.
                        The range for <number> is 1 to 1000.
                        The default value is 4.
```

OPTIONS FOR DIRECTORY-BASED QUERIES:

```
-failed              -- Show failed jobs (default)
-idle                -- Show idle jobs
-queued              -- Show queued jobs
```

RESPONSES FOR DIRECTORY-BASED QUERIES:

```
y      -- Why is the node not valid?
l      -- Show stdout.
2      -- Show stderr.
p      -- Print job information.
h,?    -- Show help.
l      -- List all jobs with the specified status.
<      -- Previous job.
>      -- Next job.
<ret>  -- Next job.
q      -- Quit.
```

Options are case insensitive and can be abbreviated to one character.

EXAMPLES:

```
% vsy 45233          -- Check job with id of 45233.
% vsy -json 45233    -- Same as above, but format output as JSON.
```

```
% vsy -jsondoc      -- Documentation of JSON format returned by "vsy -json"
% vsy !             -- Check most recent job in the current directory.
% vsy -h            -- Get this message.
% vsy -failed       -- Failed jobs in current dir.
% vsy -f            -- Same as above.
```

## List Inputs and Outputs of a Node for Navigation

### The CLI program vsx - Vov Show Inputs & Outputs

The command line program `vsx` lists the input and output files for a given node. This is useful for finding the path through the dependency graph, to support navigation along that path from the command line.

The name of this program was created because the graph depiction of a job having two inputs and two outputs looks like an X.

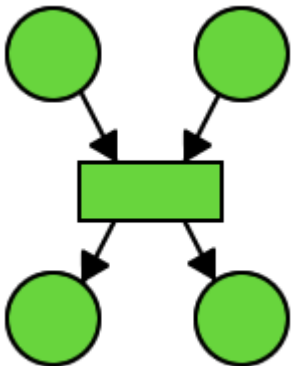


Figure 44:

### VSX

Show the inputs and outputs of nodes in the trace.

```
vsx: Usage Message

DESCRIPTION:
  Show the inputs and outputs of nodes in the trace.
USAGE:
  % vsx [options] node_spec ...

OPTIONS:
  -v                -- Increase verbosity.
  -h                -- Show this message.
  -O <format>       -- Specify output format.
  -inputs           -- Inputs only.
  -i                -- Inputs only.
  -outputs          -- Outputs only.
```

```
-o                -- Outputs only.

node_spec        can be a VovId, a file name, or a bang (!).
                  The bang means the most recent job in the
                  current working directory.

DEFAULTS:
  The default output format is:
    "@ID@ @STATUS:10@ @ORIGIN@ @DB:8@ @NAMEORCOMMAND:45@"

EXAMPLES:
  % vsx aa
  % vsx foo bar
  % vsx -O "@ID@ @LEVEL@" 00123456
  % vsx !
```

## Examples

```
% vsx bb.txt
00000364 VALID      fc----- vw cp aa.txt bb.txt
      >>>> Node 00000012  VALID  ${TOP}/bb.txt
00000392 VALID      fc----- vw cp bb.txt cc.txt
```

The argument "!" is interpreted as the most recent job in the current directory.

Example:

```
% vsx !
00000012 VALID      fc----- ${TOP}/bb.txt
00000010 VALID      ----w--  ${VOVDIR}/tcl/vtcl/capsules/vov_cp.tcl
      >>>> Node 00000392  VALID  vw cp bb.txt cc.txt
00000015 VALID      fc----- ${TOP}/cc.txt
```

## Clean Out Stuck, Unproductive Nodes

### The CLI Program vsz - Vov Stuck Node Zap

The command line program `vsz` is a general clean up utility that removes nodes in the dependency graph that get stuck in an unproductive state.

The name of this program can be thought of as the acronym for "Vov Stuck & Zapped."

### VSZ

Zap away problematic nodes.

```
vsz: Usage Message

DESCRIPTION:
  Zap away problematic nodes:
    1 - isolated files are forgotten;
```

```
    2 - blocking files that are retracing are forgotten;
    3 - other blocking files are shown.
    Optionally, check that all names in trace are canonical.

USAGE:
    % vsz [options]

OPTIONS:
    -check          -- Check Canonical names (expensive)
    -fix            -- Fix canonical names (expensive)
    -help           -- This message.

EXAMPLES:
    % vsz -help
    % vsz
```

## Checking and Fixing File Names with vsz

The command `vsz` can also report, and fix, file nodes that have names that do not follow the standard naming convention. When run with the option `-check`, `vsz` reports all the file names that are not [Canonical and Logical File Names](#). This is a computationally expensive test, requiring several minutes for graphs with thousands of files. For each non-canonical name, the utility suggests a corrected name. You can rename the file to the suggested name with CLI command `vovrename`.

The renaming is automatic if the option `-fix` is used instead of `-check`. Examples:

```
% vsz -fix... output omitted ...
```

## Examples

```
% vsz -help
% vsz
```

Isolated files are commonly created because the I/O behavior of a tool changes over time. For example, suppose you have a tool that generates an error log file because it has detected an error condition. The error log file is therefore an output of the tool. Now you fix the error condition and rerun the tool. This time the error file is not generated and is therefore no longer an output of the tool. The node associated with the error log file is still in the graph, but it has become isolated.

```
% vsz
vsz: message: No isolated node.
vsz: message: No blocking nodes that are retracing.
vsz: message: No blocking nodes.
```

## Run Tcl Scripts that Use Tcl Extensions

The command line program `vovsh` is a major component of Altair Accelerator products. It is the program which runs the Tcl scripts that provide various functions and commands in the Altair Accelerator products. It is rarely run directly by an end user but instead is run indirectly when the end user runs a command that is implemented by way of a Tcl script executed by `vovsh`.

## vovsh

The common use of `vovsh` is to execute a Tcl script which contains the Tcl extensions provided by Altair Accelerator.

```
usage: vovsh [-v] [-f script] [-anN] [-p projectName]
           [-s netinfo|procinfo|clockoffset] [-t script] [-Y Tcl
           command] [-x Tcl command]
-v:      Print version.
-f:      Evaluate a Tcl script. Additional arguments are passed via the
         'argv' and 'argc' variables to the TCL script. 'argv0' is set
         to the name of the script. The script can be specified in
         compact form, in which case it will be looked for in the
         directory $VOVDIR/tcl/vtcl/<script>.tcl.
-a:      Ignore PROCS_TO_TRACK property, track all processes when -s
         procinfo is enabled.
-n:      Do not connect to any project.
-N:      Don't send auth credentials even if available. (-t, -f)
-p:      Describe project on command line (ignored by vovsh)
-s:      Exchange network-info | processes-info | clock-offset
         information with server.
-t:      Same as -f, (deprecated, for backwards compatibility).
-Y:      Same as -x, (deprecated, for backwards compatibility).
-x:      Execute a Tcl command and exit.
```

## Examples

```
% vovsh -f some/tcl/script.tcl
% vovsh -f $VOVDIR/tcl/vtcl/vsi.tcl
```

Another useful application of the program is the execution of simple Tcl statements with the option `-x`:

```
% vovsh -x 'puts [clock format 1289361743]'
Wed Sep 09 20:02:23 PST 2024
```

## vovps

`vovps` lists all processes, similar to the UNIX command `ps`. It is useful to understand the process hierarchy of Accelerator jobs, and sometimes to find which PIDs to kill in a shell.

The `-c` and `-p` options show the child processes and parent processes respectively of a given pid. The results are a Tcl list of pid, program name, and owner.

```
usage: vovps [-an] [-c pid] [-dgsSJj] [-p pid] [-mP]
-a:      Print all processes, one per line, like a simple version of 'ps'
-n:      Only print pids: must come before -c or -p
-c:      Print the children of process <pid>. The printing is in Tcl
         format, but it is also easy to parse.
-d:      Debug process info collection
-g:      Include list of processes with same PGID to children
-s:      Include list of processes with same SID to children
-S:      Include state of process R,S,T,Z...
-J:      Include list of processes with same VOV_JOBID to children
```

```
-j:      Like -a, but show the VOV_JOBID also and LM_PROJECT instead of
        command line
-p:      Print the parents of process <pid>. Same format as for -c
-m:      Print more detailed memory usage information (Linux).
-P:      Use 'ps' program to scan the processes. This behavior can also
        be selected by setting the environment variable VOV_USE_PS.
```

## Examples

### Example 1:

```
% vovps -a | grep fire
31155 28981  1h06m  0    0    0.00    0.01 bkring  vovfire28981.cs
32381 32260  56m35s 977  219  6.45   70.33 bkring  firefox
```

### Example 2: (the shell substitutes the current pid for '\$\$'):

```
% vovps -p $$
set parents(28756) {
  28754 "xterm" cadmgr
    1 "init" root
}
```

## Search Log Files

vovsearchlog searches log files that are daily generated by server, taskers and daemons.

## vovsearchlog

Search log files.

```
vovsearchlog: Usage Message

DESCRIPTION:
  Search log files

USAGE:
  % vovsearchlog [OPTIONS]

OPTIONS:
  -h                -- This help.
  -p                -- Pattern in a regular expression.
  -d                -- The number of days starting today and backward.
                    The default is 10 days.
                    If -s and -e are specified, this is ignored.
  -start            -- Start time of log files. Format is YYYYMMDD,
                    for example, 20171020.
  -end              -- End time of log files. Format is YYYYMMDD,
                    for example, 20171025.
  -f                -- Each line starts with filename, otherwise
                    filename is printed once before searched
                    contents are displayed.
  -server           -- Search only in server logs.
```

```
-tasker          -- Search only in logs from vovtasker, tasker
                  startup and vovtaskermgr.
-daemon          -- Search only in daemon logs.
-all            -- Search from any logs.
```

EXAMPLES:

```
% vovsearchlog -p ERROR -server -d 10
% vovsearchlog -p ERROR -start 20170101 -d 30
% vovsearchlog -p ERROR -start 20170101 -end 2017101
% vovsearchlog -p 000021470
% vovsearchlog -d 2
% vovsearchlog -p "Job.*code" -d 30
% vovsearchlog -p "termination?" -d 30 -tasker
```

SEE ALSO THESE COMMANDS RELATED TO LOGS:

```
vovcompresslog, vovloghandler, vovcleanup
```

SEE ALSO THESE COMMANDS THAT SHOW INFORMATION:

```
vovjobqueue,      vovset list,      vovfind,
vel,              voveventmon,      vovshowconnection,
vovproject list,
vovtaskermgr show
vsi, vls, vst,
vovdoc
```



# Create Sets with Selection Rules

## Set Names

Every set has a name and the name must be unique.

The name of a set can be any string, including spaces. However, obeying the following rules will make it easier for you to use and browse sets:

- The maximum length of a set name is 512 ASCII characters
- Consider the set name as a "path" in which components are separated by the colon ':'.  
Examples:

```
User:joe:tmp:myset  
All:clevercopy
```

- A colon at the beginning of the name is not necessary.
- Avoid using spaces in the name.
- Names beginning with three semicolons ';;;' used to be reserved for system sets, which are **protected**. These sets cannot be forgotten or modified. The user is now allowed to create sets with such names, but we discourage it.
- Names beginning with three '@@@' or with 'tmp:' indicate special transient sets. A **transient set** does not generate attach/detach events, thus avoiding unnecessary traffic between the server and the GUI clients.

## Selection Rules

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

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

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

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

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

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

is evaluated as:

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

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


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

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

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

This rule is equivalent to:

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

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

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

## Examples of Selection Rules


Selection Rule	Explanation
<code>isjob==1</code>	Select all jobs
<code>isjob</code>	Select all jobs (equivalent to <code>"isjob!=0"</code> )
<code>!isjob</code>	Select everything except jobs (equivalent to <code>"isjob!=0"</code> )
<code>not isjob</code>	Select everything except jobs (equivalent to <code>"isjob!=0"</code> )

Selection Rule	Explanation
<code>IS_FILE and db!=FILE</code>	Select all files which are not in the database <code>FILE</code>
<code>status==RUNNING</code>	Select all nodes whose status is <code>RUNNING</code>
<code>status==RUNNING, VALID</code>	Select all nodes whose status is <code>RUNNING</code> or <code>VALID</code>
<code>status!=RUNNING, VALID</code>	Select all nodes whose status is neither <code>RUNNING</code> nor <code>VALID</code>
<code>status!=RUNNING and status!=VALID</code>	Select all nodes whose status is neither <code>RUNNING</code> nor <code>VALID</code>
<code>isjob status==INVALID</code>	Select all invalid jobs
<code>isjob duration&gt;300</code>	Select all jobs that have lasted more than 5 minutes (i.e. 300 seconds).
<code>isjob command~~aa</code>	Select all jobs with a command line containing the string <code>aa</code> ; the <code>~~</code> operator is used for case insensitive match.
<code>isjob age&lt;600</code>	Select all jobs completed less than 10 minutes ago.
<code>isjob &amp; age&lt;600</code>	Select all jobs completed less than 10 minutes ago.
<code>isjob AND age&lt;600</code>	Select all jobs completed less than 10 minutes ago.
<code>isfile name~ccc</code>	Select all files with name containing <code>ccc</code> .
<code>isjob inputs&lt;2 status==INVALID</code>	Select all jobs that have fewer than 2 inputs and are invalid.
<code>isjob AND (inputs&lt;2 OR status==INVALID)</code>	Select all jobs that have fewer than 2 inputs or are invalid.
<code>isjob (inputs&lt;2   status==INVALID)</code>	Select all jobs that have fewer than 2 inputs or are invalid.
<code>isjob inputs&lt;2   status==INVALID</code>	Select all jobs that have fewer than 2 inputs, and all nodes (including non-jobs) that are invalid. See notes on AND/OR precedence, above.
<code>isfile status==VALID age&gt;=600 name~xxx</code>	Select all valid files older than 10 minutes whose name contains the string <code>xxx</code> .

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

## Selection Rules and Incompatible Fields

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

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

For example:

```
# Look for skipped jobs
```

```
isjob ISAUTOFLOW==1
```

## Regular Expressions in Selection Rules

The operators `~`, `!~`, `~~` and `!~~`, which perform string matching, operate with regular expressions (Tcl 8.0). These expressions use the `ed` syntax. For a complete description of the syntax, invoke one of the following commands:

```
% man -s n regexp  
% man ed
```

The following is a short summary of the syntax for regular expressions.

- The period `.` stands for any character.
- The asterisk `*` stands for zero or more repetitions of the previous expression.
- The plus `+` stands for one or more repetitions of the previous expression.
- A backslash `\` can be used to escape a literal period, asterisk or plus sign.

### Examples

The regular expression `".*` matches every string (any character repeated any number of times).

If you want to make a set of all the files whose name contains `.std.`, specify the regular expression `".std\.*"`, in which `".\"` matches the period.

## Time Specifications

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

The letters are defined as follows:

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

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

## Examples of TimeSpecs

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

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

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

- past hour
- today
- yesterday
- this week
- last week
- past week
- this month
- this month full
- last month
- past month
- past 30days
- this quarter
- last quarter
- this year
- last year
- YYYY, such as 2016, would be the entire year of 2016
- YYYYMM, such as 201016, which would be the month of December in 2016
- YYYYMMDD, such as 20100116, which would be Jan 15 2016
- YYYYwWW, such as 2017w4, which would be week 4 in year 2017
- Month YYYY. Example: Sep 2017
- start-finish, where on each side of the '-' is a timestamp specification that is parsed by `VovScanClock`. Example: 20090101-20090301

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

## Advanced Information

### Tasker: vtk\_tasker\_set\_timeleft

This procedure is used in the context of time-variant tasker resources. It takes a single non-negative argument, which is interpreted as the maximum expected duration for a job to be dispatched to the tasker.

For example, if a tasker does not want to accept jobs longer than 2 minutes, it will say:

```
namespace eval VovResources {
  proc SmallJobsOnly {} {
    vtk_tasker_set_timeleft 2m
    return "@STD@"
  }
}
```

The argument of `vtk_tasker_set_timeleft` is one of:

- The string "UNLIMITED", meaning that the tasker accepts jobs of any length
- A positive integer
- A time specification
- The number 0, in which case the tasker is effectively suspended, because it does not accept any job

```
...
# -- During the day, suspend the tasker.
vtk_tasker_set_timeleft 0
...
```

### Quantization of "timeleft" and "xdur"

For efficiency of the server-tasker messaging interface, all finite values passed to `vtk_tasker_set_timeleft` are silently quantized to the next largest level according to the following table:

Time Left	Quantized value
0-60s	Round up to the next multiple of 5s
60s-30m	Round up to the next multiple of 1m
30m-1h	Round up to the next multiple of 2m
1h-3h	Round up to the next multiple of 5m
3h-6h	Round up to the next multiple of 10m

Time Left	Quantized value
6h+	Round up to the next multiple of 30m

## Troubleshoot Taskers

### Troubleshooting TASKERS on UNIX using rsh

If you cannot connect taskers to a server, check your license file with `rlmstat`. The total number of taskers you can connect to VOV servers is limited by the license.

To start a tasker on a remote machine, `vovtaskermgr` uses `rsh` as determined by `vovrsh`. However, a problem with your account setup can prevent remote taskers from starting. The following tests check if your account is setup to run `rsh` correctly:

- ```
% rsh hostname date
```

This test should give you the date without asking for a password and without any other message. If a password is required, adjust the `.rhosts` file in your home directory. If any extra message is displayed, consider making the appropriate changes in your `.cshrc` file to eliminate them.

- ```
% rsh hostname vovarch
```

This test should give you the platform name for the remote host, nothing else. If this message does not appear, verify that your `.cshrc` file is sourcing the `.vovrc` file, even for non-interactive shells. (A common trick in `.cshrc` design is to exit early when it is discovered, by testing for existence of the variable `$prompt`, that the shell is non-interactive. If the exit keyword exists in your script, move the sourcing of `.vovrc` before it.)

- ```
% rsh hostname
```

(A login message appears at this point.)

```
% vovtasker
```

(The usage message from `vovtasker` appears.)

If this test succeeds and the previous test does not, then your `PATH` variable is incorrectly set in `.login`, while it is more appropriate to set it in the `.cshrc` file. You must correct this error before you can use VOV effectively.

- ```
% vovtaskermgr TEST host1
```

This tests if `rsh` works for a list of selected hosts. A host can pass this test and still not qualify as a tasker. This can happen, for instance, if the host does not mount the file system where the project data are stored.

- ```
% vovtaskermgr START -slow
```



This shows in detail how taskers are started. The verbose output may help you debug the problem with starting taskers.

## Troubleshooting TASKERS on UNIX using ssh

Many organizations are now switching to SSH for security reasons. If you also want to use SSH instead of RSH you need to do the following:

```
% vovsshsetup
```

Then in the `taskers.tcl` file, use the option `-rshcmd ssh`:

```
vtk_tasker_set_default -rshcmd "ssh"
```

Now do the tests to see the rest of the setup:

```
% ssh hostname date  
% ssh hostname vovarch
```

## Manage Remote Taskers Without SSH/RSH Capabilities

The program `vovtsd` is a daemon written as a Tcl script that runs using the VOV `vtclsh` binary. This daemon can be used also to start various types of agents on any type of Windows or UNIX machine.

In Windows, `vovtsd` is started from the command line and then runs in a Windows `cmd` shell. Each `vovserver` connects to `vovtsd` via TCP/IP to start the `vovtasker` process.

## vovtsd

This utility listens for requests to launch taskers for various projects, but always for the same user. The requests typically come from `vovtaskermgr`.

### Usage

```
vovtsd: Usage Message  
  
VOVTSDD: Vov Tasker Service Daemon  
  This utility listens for requests to launch taskers for  
  various projects, but always for the same user.  
  The requests typically come from vovtaskermgr.  
  
USAGE:  
  % vovtsd [OPTIONS]  
  
OPTIONS:  
  -v          -- Increase verbosity.  
  -h          -- Print this help.  
  -debug      -- Generate verbose output.  
  -help       -- This message.  
  -normal     -- Start a normal daemon (for current user)
```

```
-expire <TIMESPEC>      -- Exit from vovtsd after specified time.
-user <user>             -- Specify the user that should be impersonated.
                           vovtsd computes the port number by
                           hashing the user name.
-port <n>                -- Specify port to listen to.
-requireauth <n>         -- Require key-based auth from clients.
                           Client public keys need to be set in
                           VOVTSD_USERKEYS env var if enabled.
-userkeys "<key1> <key2> ..." -- Specify a list of public keys that are
                           allowed to authenticate the client.
```

EXAMPLES:

```
% vovtsd -normal
% vovtsd -port 16666
% vovtsd -user john -port 16000
```

## Start a Remote vovtasker with vovtsd

To use vovtsd on Windows, follow these steps:

1. Start a command shell on the Windows workstation as the user who is supposed to run the taskers. See below if this user needs to be different from the user logged in on the screen.
2. Set up the shell to use VOV with the vovinit command. This sets the needed environment variables, including PATH.

```
c:\temp> \<install_path>\win64\bat\vovinit
```

3. Mount all filesystems using the appropriate drive letter; these need to agree with the values of serverdir and vovdir in the taskerRes.tcl file for the VOV project.
4. Start vovtsd, possibly using a new window. The -normal option says to use a TCP/IP port calculated from the username. You may specify the port explicitly by using the -port option.

```
c:> start vovtsd -normal
```

## Run vovtsd as a Different User

The vovtasker started by vovtsd will run as the user running vovtsd. If you need for this to be different from the user logged in at the keyboard and screen, you have several options.

On Windows 2000 and Windows XP, you can use the runas.exe command included with the operating system. For example, on Windows XP, logged in as 'user1', you could start a command shell using:

```
C:\temp> runas /user:domain-name\username cmd
```



**Note:** You may need to mount the filesystems for that user. On Windows NT, the drive letters are shared, but on later versions, each user can mount a different filesystem on a given drive letter.

## vovtsd on UNIX

### UNIX: Start a Remote Tasker with vovtsd

The program `vovtsd` can also be used on UNIX. In most cases, you may want to use built-in operating system commands like `rsh` or `ssh`.

You must change the `taskers.tcl` file to set the `-vovtsdport` variable to the port number used by your `vovtsd` daemon. For example, if `vovtsd` is running on port 16001, then your `taskers.tcl` file needs to specify the port number and a number of directories and files **as seen by the remote host**:

```
vtk_tasker_set_default -vovtsdport 16001 -vovdir $env(VOVDIR) -  
serverdir  
/home/john/vov -logfile /home/john/vov/$env(VOV_PROJECT_DIR).swd/logs/taskers/  
@TASKERNAME@/log -executable vovtaskerroot
```

The `vovtaskermgr` command tries to start a remote tasker first with `vovtsd` and then with either `rsh` or `ssh`. Which remote shell command is tried depends on which you have configured with the `-rshcmd` option of `vtk_tasker_define` in the `taskers.tcl` file.

# Troubleshooting

## Nothing is Running

Maybe you do not have tasker connected to the project. Check the Tasker page. You will need to have administrator privileges to start the taskers.

## A Particular Job is not Running

If you try to run a job and nothing happens, a number of things may be going on:

- The job may be blocked because some of its inputs are not VALID and FlowTracer does not know how to make them VALID.
- The job requires resources that are not available. You need to check:
  - The resources requested by the job; the page associated with the job has a URL of the form /node/<jobid>.
  - The resources map
  - The resources offered by the taskers

## The Job Runs But Fails

The job page typically contains information about the reason for the failure. Go to the job page and then check the job outputs. In many cases, the reason for the failure is spelled out in either stderr or stdout.

## The Job Runs but Fails with No Information

- Check the taskers log.
- The working directory for the job may have insufficient "write" permission (typically the stderr and stdout log files are written in the working directory of the job)

## The Job Complains about a Missing File, but the File Exists

This problem is typically related to a faulty setting of the equivalence rules.

# Troubleshooting the Retrace

If retracing does not make forward progress toward completion (that is, if you repeatedly request a run and VOV executes the same jobs), you probably have an incomplete flow. This could be caused by a number of reasons.

A job will only be executed if all inputs are VALID and if all outputs are either INVALID or MISSING. If an input for the job is also a primary input and is INVALID or MISSING, then the job is blocked. You must use one of the following options:

- If you think the job should be executed now, simply execute it manually, for example with `vovfire`. If the job needs an input which is invalid, you will get an explicit warning. In this case, first solve the problem of validating the input.
- Remove the dependency on an input that is INVALID if that input is not required for the job. Edit the input list of the job using the navigation dialog, disconnect the inputs that are not VALID, then request the retracing.

The situation where the output of a job is VALID and the job itself is not, occurs either because some jobs have not been run or because someone or something has changed the output file without VOV's knowledge. VOV cannot distinguish between the two cases and refuses to execute the job because it would overwrite the output and lose all changes. If you want the job to be executed again, you can:

- Run the job manually (for example, `vovfire jobId`)
- Ask VOV to bring the flow in a consistent status with `vovproject sanity` and then request a run.

# Create a FlowTracer Project

As was mentioned earlier, FlowTracer is "project oriented". Consider all the jobs that work together to reach a goal, and all the files that are inputs to the jobs, or their outputs. If these are to be managed by FlowTracer, then you should give them a "project name", and register the project with FlowTracer by referring to that name when creating a FlowTracer project.

Examples of projects:

- Compiling a program on one platform
- Compiling a program on N platforms
- Running regression for a software product
- Designing a chip or some sub-blocks of a chip

## Also in this Section

## Project Creation Overview

An overview of setting up a new project:

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

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

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

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

```
% vovproject enable project
```

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

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

## Each Project Needs a Server Host

When you create a project, you start up the FlowTracer system which runs an instance of the vovserver program. You should pick a host to run the vovserver program that has resources to run the system efficiently.

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

- The host has access to all the jobs (executables) and their input and output files, directly or through network filesystems (such as NFS).
- The host has adequate main memory: at least 64MB of RAM for traces with 20,000 files in the trace -each node in the trace uses about 1kB of RAM.

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


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

## Project Creation Command

To create a new project, use the `vovproject create` command.

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

| Option       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |              |             |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|-------------|
| -type        | <p>Select the project type which influences only the creation of a new project; the initialization of the <a href="#">project configuration</a> files. These files control the project environment, tasker machines, users, security, etc., and are copied from the project type directory to the configuration files of the newly-created project.</p> <p>The standard project types are usually augmented by the site administrator to include ones specific to your site's work environment. This reduces the need for users to configure their own projects. The types are defined as subdirectories in <code>\$VOVDIR/local/ProjectTypes</code>.</p> <p>If no specific type is named, the new project will receive the template configuration files from the <code>generic</code> type. The following standard project types are available:</p> <table><tr><th>Project Type</th><th>Description</th></tr></table> | Project Type | Description |
| Project Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |              |             |

| Option             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | <code>generic</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | The default type. Specialized for FlowTracer.                                                                                                                                                                                                               |
|                    | <code>licmon</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Specialized for Monitor. Only use <code>lmmgr start</code> to create a Monitor project. This LM-specific management command also checks that auxiliary daemons and scripts are up-to-date. The <code>vovproject</code> command does not check.              |
|                    | <code>vnc</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Specialized for Accelerator. Use <code>ncmgr</code> to create a Accelerator project because it also creates the necessary entry in the <code>NC_CONFIG_DIR</code> directory (usually <code>\$VOVDIR/local/vncConfig</code> ) and sets up the jobs database. |
|                    | <code>licadm</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Specialized for an older form of licensing based on vovserver, now replaced by ALM.                                                                                                                                                                         |
|                    | <p>More types can be added by creating new subdirectories and by modifying the template configuration files in the directory for that project type. Start a new project type by copying the configuration files from the "<code>\$VOVDIR/./common/etc/ProjectTypes/generic</code>" type, and change them to suit the needs at your site.</p> <p>For example, you might create a <code>Simulation</code> type, or a <code>PhysicalDesign</code> type, with specific tasker machines.</p>                                                                                                                                                                                   |                                                                                                                                                                                                                                                             |
| <code>-port</code> | <p>Choose a port number for the project. It is required that each project running on a host has a unique port; this is an operating system restriction. By default, the port number is computed automatically by hashing the project name into the range 6200-6255. It is possible for two project names to map to the same port. Such port collisions are handled by searching upward in the registry (starting at 7200) until an open port is found.</p> <div>  <b>Note:</b> To run multiple projects concurrently on the same host, ensure they use different ports.         </div> |                                                                                                                                                                                                                                                             |
| <code>-dir</code>  | <p>Choose the directory where you want to store all administration files for the project. The directory in which the vovserver program is launched is</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                             |



| Option        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <p>called the <i>server working directory</i>. The server working directory contains the control database as well as all the project specific files.</p> <p>By default, the server working directory is ~/vov on UNIX and \$VOVDIR/swd on Windows. For very important projects, the server working directory is explicitly set in an area where it can be subject to the same revision control methods as all the rest of the design data. In such cases, the directory is often called vovadmin.</p> <p>You also need to consider the canonical name of the server working directory. For further information, refer to <a href="#">Canonical and Logical File Names</a>.</p> |
| <projectname> | <p>Choose a name for the project. The project name can be any alphanumeric string, such as: cpu, badge, MicroProcessor, mike, xyz99. The name is case sensitive. Special characters, underscores "_" and dashes "-" are allowed.</p> <p>The name of the project and the host must each be 100 characters or less.</p> <p>The following strings are not allowed as project names: "unknown" "none".</p> <p>The project name is used to compute the default <a href="#">port number</a> used by the vovserver program to listen for connections. See above for restrictions on the port number.</p>                                                                              |

Below is an example of creating a project using the default port number, the default type `generic`, and the default working directory:

```
% vovproject create test
Creating a new project:
Directory /Users/designerabc/vov
Type      generic
Name      test
Port      automatic
vovproject 11/11/2015 07:28:30: message: Creating directory /Users/designerabc/vov/
test.swd/.
vovproject 11/11/2015 07:28:30: message: Created setup file '/Users/designerabc/vov/
test.swd/setup.tcl'
vovproject 11/11/2015 07:28:30: message: Copying initial DB config file...
vovproject 11/11/2015 07:28:36: message: Starting a VovServer
for project      test@mac05.local
PORT=automatic
vovserver Nov 11 07:28:36 VOV server MacOS/2015.11 Nov  5 2015 09:17:16
Copyright © 1995-2025, Altair Engineering
vovserver Nov 11 07:28:36 Version MacOS/2015.09 Nov  5 2015 08:55:57 Copyright ©
1995-2025, Altair Engineering
vovserver Nov 11 07:28:36 Project was started on: Thu Nov 11 07:28:36 2015
vovserver Nov 11 07:28:36 The server port is 6392
vovserver Nov 11 07:28:36 Running with capacity 256
vovserver Nov 11 07:28:36 ##### READY TO RECEIVE port=6392 #####
vovserver Nov 11 07:28:36 The server pid is 41613
vovserver Nov 11 07:28:36 << Redirecting output  /Users/designerabc/vov/test.swd/
logs/server.2015.11.11.log >>
```

## Start the vovserver

There are two ways to start the vovserver:

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



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

## vovproject

The vovproject tool is used to manage a VOV project.

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

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

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

USAGE:
  vovproject [command] [command arguments]

COMMANDS:
  archive           # Archive projects.
  unarchive         # Unarchive projects.
  backup            # Make a backup copy of the trace
  create            # Create a new project
  destroy           # Completely remove all server data
  enable            # Enable the shell to connect to the project
  info              # Get info and stats about the project
  list              # List all the projects
  rename            # Change the name of the project
  report            # Print reports about the projects
  reread|reset|refresh # Re-read all configuration files
  restore           # Restore the trace from a backup file
  sanity            # Check internal consistency of server database
  save              # Save trace database to disk
  start             # Start a project server
  stop              # Stop a project server

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

## vovserver

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

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

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

## Connect to a Project via the Command Line

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

Example:

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

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

The project is now set up for interaction. Example:

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

## Troubleshooting: Cannot Enable Project

Problem: you run the following command:

```
% vovproject enable MYPROJECT
```

and you receive an error similar to this:

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

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

Troubleshooting tips:

```
% vovproject list -a
```

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

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

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

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

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

## Shutting Down a Project

### Shut Down a Project

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

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

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

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

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

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

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

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

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

### Last-resort vovserver Shutdown Method

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

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

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

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

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

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

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

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

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

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


### Automated Script Calls for vovserver Shutdown

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

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

Additional information:

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

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

## Destroy a Project

A project must be stopped before it can be destroyed.

When a project is destroyed:

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

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

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

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

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

## Sanity Check for vovserver

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

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

```
% vovproject sanity
```

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

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

```
% vovproject reread
```

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

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

- Checks for a license



## Resource Management

Altair Accelerator includes a subsystem for managing computing resources. This allows the design team to factor in various constraints regarding hardware and software resources, as well as site policy constraints.


This mechanism is based on the following:

- Resources required by jobs
- Resources offered by taskers
- Resource maps, as described in the file `resources.tcl`

There are several types of resources, which are listed below:

| Resource type                                     | Representation             | Explanation                                                                                                                                                                                                                                                   |
|---------------------------------------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Job Resources                                     | <code>name</code>          | A resource required by a job. If the quantity is not shown, the default is 1; "unix" is the same as "unix#1".                                                                                                                                                 |
| Uncountable Resources<br>(also called Attributes) | <code>name</code>          | These resources represent attributes of a tasker that are not countable. For example, a tasker may have attributes such as "unix" or "linux". The quantity is not shown for these resources and it defaults to MAXINT; "unix" is equivalent to "unix#MAXINT". |
| Quantitative Resources                            | <code>name#quantity</code> | Example: The resource <code>RAMTOTAL#2014</code> on a tasker indicates the total amount of RAM on that machine. On a job, it says that the job requires at least the shown amount of RAMTOTAL.                                                                |
| Consumable Resources                              | <code>name/quantity</code> | Example: <code>RAM/500</code> assigned to a job indicates that the job consumes 500 MB of the consumable resources RAM.                                                                                                                                       |
| Negated Resources                                 | <code>!name</code>         | Example: "unix !linux" on a job indicates that the job requires a UNIX machine but not a Linux one.                                                                                                                                                           |

The definition of the quantity is related to the context of the resource. If the context is a tasker, quantity represents how much of that resource is available from the tasker. If the context is a job, quantity represents how much of that resource is required by the job.


 **Note:** Negated resources are allowed only for the context of a job.

The *unit of measure* is determined by convention for each resource. For example, the resource `RAMTOTAL` is measured in MB. By default, quantity is assumed to be 1; the notation `foo` is equivalent to `foo#1`.

A *resources list* is a space-separated list of resources, which are typical resources offered by the taskers. The following example indicates that a job requires at least 128 MB of RAM and a UNIX host, but not a Linux host.

```
RAMTOTAL#128 unix !linux
```

A *resources expression* is a space separated list of resources and operators: typical resources requested by the jobs or mapped in the resource map set. Operators can be one of the following: `<blank space>`, `&`, `|`, `OR`, `AND`, `!`, and `NOT`. The operators are defined in the table below.

 **Note:** Logical `AND` has precedence over logical `OR` operations.

| Operator                         | Description               |
|----------------------------------|---------------------------|
| <code>&lt;blank space&gt;</code> | implicit logical AND      |
| <code>&amp;</code>               | explicit logical AND      |
| <code>AND</code>                 | explicit logical AND      |
| <code> </code>                   | explicit logical OR       |
| <code>OR</code>                  | explicit logical OR       |
| <code>!</code>                   | explicit logical negation |
| <code>NOT</code>                 | explicit logical negation |

For example, a job may have the following resource requirements:

```
RAMTOTAL#128 unix !linux | RAMTOTAL#512 & linux
```

This job requires either a UNIX host with at least 128 MB of RAM, but not a `linux` host or a Linux host with at least 512MB of RAM.

## Use of Tasker Lists in Resources

The recommended use case is to have one tasker list reference as the first resource in the list. This narrows the scope of the scheduler and improves performance which is the main goal of tasker lists.

Use of tasker lists within resource expressions is not recommended. In particular the use of `OR` operators with tasker lists can result in unexpected behavior.

# Resource Monitoring

Table 1: Summary of Methods to Monitor Resources

|         |                                                                                               |
|---------|-----------------------------------------------------------------------------------------------|
| CLI     | <code>vsm -r</code><br><code>vsm -panel resmaps</code>                                        |
| GUI     | In the VOV Monitor dialog, click on the Resources tab                                         |
| Tcl     | <code>vtk_resources_get (old)</code><br><code>vtk_generic_get resourcemaps array (new)</code> |
| Browser | Visit the Resources page                                                                      |

## Rename Nodes

Renaming of nodes is a task for more advanced users. In most cases, it is simpler to forget all nodes and rebuild the trace from scratch.

You do not need to continue reading this section unless you really want to learn about renaming.

Occasionally, in the course of a design, directories get moved or renamed. Rather than forgetting the work done in the old directory and redoing it in the new directory, it is possible to tell VOV to rename the nodes in the trace.

### Renaming Using the Command Line Interface (CLI)

The utility to change names is `vovrename`.

```
vovrename: Usage Message

USAGE:
  vovrename [-set <name>] [-n] <old> <new>

OPTIONS:
  -n                -- Show effect of substitution, do not do it.
  -set <name>       -- Restrict substitution to given set

EXAMPLE:
  % vovrename abc xyz
```

To affect the whole trace, or to affect only the elements in a given set use:

```
% vovrename -set setname oldstring newstring
```

This command affects the:

- Name of files
- Command line of jobs
- Working directory of jobs

For `oldstring` and `newstring`, try to use long strings. If you use short strings, the change might affect more nodes than you want. An example to avoid would be the following command:

```
% vovrename a b   ### NEVER DO ANYTHING LIKE THIS!
```

which replaces all a's into b's with rather dramatic effects on the usability of the trace.

### Example

In this example, you have built a CHIP in a directory `/users/mydesign/work` and have decided to move all the files in another directory by doing the following:

```
% cd /users/mydesign
% mv work reallygoodchip
```

FlowTracer now is in a state of uncertainty: it knew about some files in the old directory, now it cannot find them.

- First you need to get the logical canonical names of the two directories

```
% set oldname = `vovequiv -p /users/mydesign/work`  
% set newname = `vovequiv -p /users/mydesign/reallygoodchip`
```

- Then you can use `vovrename` to inform FlowTracer that some files have been moved:

```
% vovrename $oldname $newname
```

## Miscellaneous

### Enable CLI Access on UNIX

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

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

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

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

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

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

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

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

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

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

## Verify Access to Altair Accelerator Products

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

1. At the command prompt, enter the following:

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

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

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

```
% vovarch
```

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

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

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

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

or

```
% vovarch
```

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

## Enable Altair Accelerator for Non-Interactive Shells

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

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

Early exit from non-interactive shell login CSH script:

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

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

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

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

Source helper file before exit in login CSH script:

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

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

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

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



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

## Troubleshooting the UNIX Setup

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

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

```
% csh -c vovarch
```

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

## Enable CLI Access on Windows

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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


## Verify Context Is Working

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

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

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

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

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

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

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

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

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

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

# FlowTracer For UNIX Users

## FlowTracer Functionality

Consider a particular situation where obtaining a result involves running a group of steps called jobs that each run a program to perform an activity. Imagine that each job may build on what other jobs have done, and each job has a proper time to run in the sequence of jobs. Some jobs have dependencies on other jobs having been run, and may in turn enable other jobs to run that depend on them. The full sequence of jobs needs to be run in order to obtain the result.

The work to schedule and dispatch the jobs can become too difficult to manage by hand as the number of jobs increase and the rules of dependency get complicated.

FlowTracer is a product that can manage such a situation to schedule and run the jobs so that all dependencies are enforced and the minimal amount of work is performed to obtain the desired result.

Although the work of FlowTracer involves the scheduling and dispatching of jobs which are computer processes and which use file system resources, this section will not be concerned with those processes. This section will be concerned with the processes that implement the FlowTracer system, which perform the scheduling and dispatching.

## Command Line Programs

FlowTracer is implemented with a client/server architecture. The server component is provided by a set of background daemon processes that work together to provide functionality. These daemon processes are started and stopped as desired.

One part of the client component is a set of programs that can be run from the command line to perform specific actions. A few of these will be covered in this section.

The User Interface section of this chapter will cover the full set of client components.

When you have a command line context that is enabled to run Altair Accelerator programs, you can start the FlowTracer system for the first time. This will start up a background daemon processes that provide the FlowTracer functionality. You will do this below, but first, you will learn about three command line programs and the concept of a project.

FlowTracer provides a rich set of interactive programs that can be used from the command line to control the product. These are called the Command Line Interface (CLI). These correspond to Chapter 1 commands in UNIX. They are programs that can be run from the command line or from shell scripts. You call them to perform a specific task to accomplish a desired goal. They are controlled and adjusted by command line parameters,

The Altair Accelerator programs have a naming convention of using the prefix 'vov' in front of a root subject name ('vovproject', 'vovconsole', 'vovarch', ...). This convention is used so that the program names do not clash with names of programs from another product. There are exceptions to this naming convention ('vsr', 'vsi', ..) where the program names are acronyms for phrases having 'vov' as the first word. The use of 'vov' as the naming convention comes from an Italian liquor named 'Vov'. A very early prototype of the system was started at a place and time when projects were named after liquors, and the Italian liquor "Vov" chosen as the name for the project.

Most of the CLI command programs interact with a running FlowTracer system to interact with it and manage it, or report about it. They are most useful when FlowTracer is actively running. A few CLI

programs interact with the FlowTracer system, even when it is not running. These programs manage the static parts of the product ('vovversion', 'vovproject', 'vovarch', 'vovdoc',...), and are useful even when FlowTracer is not actively running.

## Verify Access to FlowTracer

You can verify that your command line environment is set correctly to access FlowTracer by running one of the interactive programs that does not require that FlowTracer be running. A good one to use for verification is the CLI command `vovarch`, which reports the machine architecture you are using.

Example of simple command to verify access to FlowTracer:

```
% vovarch
linux64
```

You can also ask the system to report on all the projects that have been set up within FlowTracer using the CLI command `vovproject` with the option `list`. If you are doing this for the first time here, running `vovproject list` will show an empty list. If you run `vovproject list` at a later time, you'll see whatever projects exist then.

Example of empty list of projects:

```
% vovproject list
PROJECT  OWNER  HOST  PORT  STATUS`
```

The `vovproject` CLI command is a work-horse command in the FlowTracer system. There are many options for the command that qualify what project operation it should do. It is like many UNIX commands in that it has a command line option to ask for help on its usage.

View usage message from `vovproject`:

```
% vovproject -h
:
```

This is an example of asking for help that holds true for most FlowTracer CLI commands. The `-h` parameter is a request to output a usage statement.

Copyright © 1995-2025, Altair Engineering    [www.altair.com](http://www.altair.com)

### DESCRIPTION:

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

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

### USAGE:

`vovproject [command] [command arguments]`

### COMMANDS:

|                        |                                     |
|------------------------|-------------------------------------|
| <code>archive</code>   | # Archive projects.                 |
| <code>unarchive</code> | # Unarchive projects.               |
| <code>backup</code>    | # Make a backup copy of the trace   |
| <code>create</code>    | # Create a new project              |
| <code>destroy</code>   | # Completely remove all server data |

```
enable      # Enable the shell to connect to the project
info        # Get info and stats about the project
list        # List all the projects
rename      # Change the name of the project
report      # Print reports about the projects
reread|reset|refresh # Re-read all configuration files
restore     # Restore the trace from a backup file
sanity      # Check internal consistency of server database
save        # Save trace database to disk
start       # Start a project server
stop        # Stop a project server
```

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

As a UNIX user, you may be curious about the processes that run when the FlowTracer system is active. Use the UNIX `ps -f` command to see what is running before the FlowTracer system becomes an active one so you can have a baseline view to compare to what you'll see later when FlowTracer is running.

When FlowTracer is actively running, it is implemented as a group of programs running in the background as services. The group of programs cooperate with each other to perform the FlowTracer function. After getting FlowTracer running, use `ps -f` to view the set of UNIX processes that are running to implement the functionality.

## FlowTracer is Project Oriented

The FlowTracer system is "project oriented". A user establishes a project for each use case where there is a goal that is obtained by running a large set of jobs having rules of dependencies. The large set of jobs and their dependencies make up a project that FlowTracer manages. Jobs work with files, using them as input, and producing files as output. The files in the project are stored in the file system structure within the project folder.

A FlowTracer project is the focus of all work that FlowTracer does. This is why most the CLI commands need a project context to target their actions to a given project. You should always be thinking about the project context whenever you do any actions with FlowTracer.

For instance, FlowTracer is started and stopped by starting and stopping a project. This matches the underlying process architecture. Each project has a separate group of FlowTracer daemon processes that work together to take care of that project.

To get FlowTracer started for the first time, you request that a FlowTracer project be both created and started. Subsequent requests to start a project will simply be to start it. Only the first time involves the create step. It is possible to create a project, and not start it at that moment.

You create a project and default to have it start running with the program `vovproject` using the option `create` with no other control parameters.

After running that command, you can see that the project exists and is running. Here are two imaginary sequence of commands to demonstrate how `vovproject list` would eventually show a project after it is created and started. In the next section below, there is an actual example to try.

Imaginary commands showing a project listed after creating it:

```
% vovproject create myNewProject > create.log
% vovproject list
  PROJECT      OWNER   HOST      PORT  STATUS`
1  myNewProject bill   unixhost  15918  running
```

```
% ps -f > ps.log
```

Imaginary commands showing a project created but not started right away:

```
% vovproject create -nostart myNewProject > create.log
% vovproject list
  PROJECT      OWNER   HOST      PORT  STATUS`
1  myNewProject bill   unixhost  auto* never started
% vovproject start myNewProject > start.log % vovproject list
  PROJECT      OWNER   HOST      PORT  STATUS`
1  myNewProject bill   unixhost  6252  running
% ps -f > ps.log
```

## Start FlowTracer by Creating a Project

This exercise is a way to see the simplest case of FlowTracer running. There is no productive work involved in the example project. This means that FlowTracer is not being asked to do the work it would normally do with regard to accepting a set of programs to schedule and dispatch based on dependency rules.

The example simply runs the FlowTracer product with the intent to show the framework, not to get something done with it. This discussion will not include detailed explanation about options in the commands, or the full story of exactly what happens in the background.

A user of FlowTracer would normally establish a working directory to be the top level directory of a project. When doing something useful, the project directory structure would hold the files that support the work. For this example, there is no work planned for the project, so the project directory structure will have nothing in it.

A convenient way of working is to use the project name as the name of the top level working directory. For this exercise, imagine that the project name is "see-it-run" to indicate the simple ambition of the project. Examples will be as if your user name is "bill" and that the top level working directory is in your home directory (/Users/bill) and it will be named after the project (/Users/bill/see-it-run).

Session activity to try - Create a project:

```
% cd
% mkdir see-it-run
% cd see-it-run
% vovproject create see-it-run
Creating a new project:
  Directory /Users/bill/vov
  Type      generic
  Name      see-it-run
  Port      automatic
vovproject 09/11/2015 15:40:23: message: Creating directory /Users/bill/vov/see-it-run.swd/.
vovproject 09/11/2015 15:40:23: message: Created setup file '/Users/bill/vov/see-it-run.swd/setup.tcl'
vovproject 09/11/2015 15:40:23: message: Copy all files from /opt/rtda/201509/linux/etc/ProjectTypes/generic
vovproject 09/11/2015 15:40:23: message: Starting a VOV server
  for project  see-it-run@unixhost
  PORT=automatic
vovproject 09/28/2015 15:40:23: message: Moving old resource logs to new location
vovserver(16372) Sep 11 15:40:23 VOVserver linux/2015.09 built:Sept 10 2015 17:41:32
vovserver(16372) Sep 11 15:40:23 Copyright © 1995-2025, Altair Engineering
```

```
vovserver(16372) Sep 11 15:40:23 Checking ServerInfo lock on file ./see-it-run.swd/
serverinfo.tcl
vovserver(16372) Sep 11 15:40:23 The ServerInfo file is not locked
vovserver(16372) Sep 11 15:40:23 The server port is 6401
vovserver(16372) Sep 11 15:40:23 Sanity: Version linux/2015.09 built:Sept 10 2015
17:41:32
vovserver(16372) Sep 11 15:40:23 Sanity: Copyright © 1995-2025, Altair Engineering
vovserver(16372) Sep 11 15:40:23 Setting product name from project name see-it-run ->
ft
vovserver(16372) Sep 11 15:40:23 Changing seatlic to 'seat_ft_l'
vovserver(16372) Sep 11 15:40:23 Server metrics not enabled (flags=0)
vovserver(16372) Sep 11 15:40:23 Running as product ft
vovserver(16372) Sep 11 15:40:23 Project was started on: Fri Sep 11 15:40:23 2015
vovserver(16372) Sep 11 15:40:23 Running with capacity 256 file descriptors
vovserver(16372) Sep 11 15:40:23 ##### PRIMARY SERVER LISTENING ON PORT 6401
#####
vovserver(16372) Sep 11 15:40:23 The server pid is 16372
vovserver(16372) Sep 11 15:40:23 << Redirecting output /Users/bill/vov/see-it-
run.swd/logs/server.2015.08.28_154023.log >>
```

Now the system is managing the "see-it-run" project.

You can list the projects by running the CLI command `vovproject list`.

Session activity to try - Show list of projects after creating a project:

```
% vovproject list
PROJECT OWNER HOST PORT STATUS
1 see-it-run bill unixhost 6401 running
```

Note that the PORT number reported is used by the web server in FlowTracer as it waits for HTTP requests to open up the browser visual console.

## Background Processes of FlowTracer

Use the `ps -f` command to see the group of background processes that are running now that the FlowTracer system is started. Notice the `server pid` that was reported in the output during the start project step. The running processes are the ones that implement the FlowTracer service, one group of processes per project.

Processes that manage a running project:

```
PID PPID CMD
15981 15980 -bash
16372 1 vovserver -p automatic -jsb see-it-run
16394 1 vovsh -f /opt/rtda/201509/linux/tcl/vtcl/vovresourced.tcl -l see-it-
run
16395 1 vovsh -n -f /opt/rtda/201509/linux/scripts/vovdailylog ./
vovresourced.@DATE@.log
16421 1 launchctl bsexec / vovtasker -p see-it-run -h unixhost -r @STD@
-l ./see-it-run.swd/logs/taskers/localhost/20150911T122253/log
-I - -i 1 -V - -R - -G - -S -
-a localhost -g g1 -e - -b FULL -L - -C -1 -D 5 -T -1 -U 60
-n 0 -c 1 -P -1 -M -1 -s
16422 16421 vovtasker -p see-it-run -h unixhost -r @STD@
-l ./see-it-run.swd/logs/taskers/localhost/20150911T122253/log
-I - -i 1 -V - -R - -G - -S -
-a localhost -g g1 -e - -b FULL -L - -C -1 -D 5 -T -1 -U 60
-n 0 -c 1 -P -1 -M -1 -s
```



You can see that a series of programs have been started, and have become child processes of the system (PPID is "1"). These are the background daemon processes that manage the project.

The program `vovserver` is the main daemon process. It is run with a parameter that tells it to manage the project `see-it-run`. It works with the other daemon programs to coordinate their activity.

One other program is `vovtasker`, the dispatch handler. This process is used to run the jobs that are dispatched by the `vovserver` program as part of its work to schedule jobs and dispatch them as needed. In this simple case, the default configuration is used which sets up a single dispatch handler on the host machine so that dispatched jobs are run locally. That is why the `vovtasker` program is running on the local machine.

If you set up multiple hosts to provide resources to run jobs, you can register them with FlowTracer as additional tasker hosts where a `vovtasker` program can be running as part of the FlowTracer daemon process team.

Note that a few processes are run by calling the program `vovsh` to interpret a Tcl script file. The Tcl script provides the processing logic that implements the action for the process. One of those processes monitors activity and writes a log file to report on events.

## Working Directory for Projects

You can look in your home directory and see that it contains a directory named `vov`. This is the default location where information about the state of the FlowTracer system is kept. When you create a project, a working directory is created having a default location within the `~/vov` directory. A different location for the project's working directory can be set when creating a project. You have created a project `see-it-run` using the default, which leads to the creation of the project working directory `~/vov/see-it-run.swd`.

This project directory holds state information about the `see-it-run` project. There are directories within `~/vov/see-it-run.swd` that hold log files. From the output of the `ps` command, you may notice a command line parameter referring to the path `../logs/tasker/localhost/nnnnn/log`. This is where a log file will be written by the `vovtasker` program.

## Stop FlowTracer by Stopping a Project

You can stop the `see-it-run` project now, and put it into a stopped state. This shuts down this instance of the FlowTracer service. The stopped state of the project is stored in the `~/vov/see-it-run.swd` folder in such a way that it can be started again when requested. Again, because FlowTracer is project oriented, you stop it by requesting to stop the project using the `vovproject` CLI command with the `stop` option and the name of the project to stop.

This stop request has a confirm step to prevent accidentally stopping a project. First you run the `vovproject stop [project-name]` command, then after you qualify as having privileges to stop the project, you are prompted to confirm your request with a reply of a full "y-e-s".

Session activity to try - Stop a running project:

```
% vovproject stop see-it-run
vovproject 08/28/2015 16:18:58: message: Checking privilege to stop project 'see-it-run'
Shut down see-it-run (yes/no)? yes
vovproject 08/28/2015 16:19:05: message: Stopping all retracing
```

```
vovleader(96135) Aug 28 16:19:05
vovproject 08/28/2015 16:19:05: message: Retracing stopped
vovproject 08/28/2015 16:19:05: message: Stopping all running jobs
vovproject 08/28/2015 16:19:05: message: Running jobs stopped
vovtaskmgr 08/28/2015 16:19:05: message: Stopping all 1 taskers.
vovleader(96140) Aug 28 16:19:08 Server see-it-run killed without question
```

Even though the FlowTracer project has been stopped, and the background processes have stopped running (Did you try `ps -f` after stopping the project?), you may still request a list of projects, as you did earlier. Listing projects does not require the FlowTracer system to be actively running.

Session activity to try - List existing projects, expecting it to be stopped:

```
% vovproject list
PROJECT    OWNER    HOST    PORT STATUS
1    see-it-run bill unixhost 6401 stopped
```

This time, the request to list projects shows the one just created and shows it in the stopped state. `vovproject` is a CLI command that runs fine even when FlowTracer is not started.

If you run the `ps -f` command to see running processes, you will see the main programs are no longer running. The event monitors and loggers are still running. They do not stop immediately. They will stop after a short time if the project is not started up again.

## Start FlowTracer by Starting a Project

You can start the existing `see-it-run` project again with a simple start command.

Session activity to try - Start a stopped project:

```
% vovproject start see-it-run > start.log
```

The redirect of the output to a log file is shown here to reduce the clutter of what is shown. In normal use, the output would not be redirected to a log and you'd see it interactively. The output is similar to what is shown during the request to create project that is shown above.

You can list the status of projects on the server in the same way as before and can see that the project has been started and is now running. Check out the processes that have started by running the `ps -f` command.

Session activity to try - List existing project, expecting it to be running:

```
% vovproject list
PROJECT    OWNER    HOST    PORT STATUS
1    see-it-run bill unixhost 6401 running
```

This section gave you the basic experience of managing an installed FlowTracer system after you set up a proper shell environment. You can understand that CLI commands are like all UNIX commands - they are programs that you run to do a particular thing. Read the *VOV Subsystem Reference Guide* as a UNIX command reference guide to discover more about what each command does.

By working through this section, you have now seen the basic aspects of FlowTracer. You can verify that your shell environment is correct to access FlowTracer. You can start and stop the FlowTracer system by starting and stopping projects. You can see the processes that are running to implement an actively

running FlowTracer system. You can find the directories in the file system that hold project logs and project state.

# Reports

## Get Status of a Project

The quickest way to get a status of a project is to use `vsi`.

```
vsi: Usage Message

DESCRIPTION:
  Print basic information about a project.
  Also useful to test whether a VOVserver
  is running or not.

USAGE:
  % vsi [options]

OPTIONS:
  -help                -- This message.
  -width <N>          -- Size of command line strings (default 50).
  -j                  -- Show the most recent 6 jobs of mine.

EXAMPLES:
  % vsi
  % vsi -j
  % vsi -width 100
  % vsi -w 100
```

## Get a Report About a Project

Another method to get a report for a project is with the `report` subcommand of `vovproject`. By default, this command computes a report for the whole trace, i.e. the set of all nodes, which is called "System:nodes"

Example:

```
% vovproject report
Project luccal@lucca report on Thu Aug 13 09:42:17 PDT 2015
Set System:nodes
  Jobs      : 2
    valid   : 2
  invalid   : 0
    failed  : 0
  not run   : 0
  Job Time  : 2s
    average : 1s
    longest : 1s
First Job Start: Thu Aug 13 02:23:28 PDT 2015
Last Job End:   Thu Aug 13 02:23:30 PDT 2015
Elapse Time:   2s

      Files: 6
No FAILED jobs.
No INVALID jobs.
The 2 longest jobs in set 'System:nodes'
  VALID    1s      vw cp bb.txt cc.txt
  VALID    1s      vw cp aa.txt bb.txt
```

```
...more output omitted...
```

You can also request a report for a specific set.

```
% vovproject report Some:set:name  
... output omitted...
```

# Legal Notices

# Intellectual Property Rights Notice

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

Copyright ©1986-2025 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of the intellectual property rights held by Altair Engineering Inc. or its affiliates. Software, other products, and materials of Altair Engineering Inc. or its affiliates are protected under laws of the United States and laws of other jurisdictions.

In addition to intellectual property rights indicated herein, such software, other products, and materials of Altair Engineering Inc. or its affiliates may be further protected by patents, additional copyrights, additional trademarks, trade secrets, and additional other intellectual property rights. For avoidance of doubt, copyright notice does not imply publication. Copyrights in the below are held by Altair Engineering Inc. or its affiliates. Additionally, all non-Altair marks are the property of their respective owners. If you have any questions regarding trademarks or registrations, please contact marketing and legal.

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

## **Altair HyperWorks®, a Design & Simulation Platform**

**Altair® AcuSolve®** ©1997-2025

**Altair® Activate®** ©1989-2025

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

**Altair® Battery Damage Identifier™** ©2019-2025

**Altair® CFD™** ©1990-2025

**Altair Compose®** ©2007-2025

**Altair® ConnectMe™** ©2014-2025

**Altair® DesignAI™** ©2022-2025

**Altair® DSim®** ©2024-2025

**Altair® DSim® Cloud** ©2024-2025

**Altair® DSim® Cloud CLI** ©2024-2025

**Altair® DSim® Studio** ©2024-2025

**Altair® EDEM™** ©2005-2025

**Altair® EEvision™** ©2018-2025

**Altair® ElectroFlo™** ©1992-2025  
**Altair Embed®** ©1989-2025  
**Altair Embed® SE** ©1989-2025  
**Altair Embed®/Digital Power Designer** ©2012-2025  
**Altair Embed®/eDrives** ©2012-2025  
**Altair Embed® Viewer** ©1996-2025  
**Altair® e-Motor Director™** ©2019-2025  
**Altair® ESAComp®** ©1992-2025  
**Altair® expertAI™** ©2020-2025  
**Altair® Feko®** ©1999-2025  
**Altair® FlightStream®** ©2017-2025  
**Altair® Flow Simulator™** ©2016-2025  
**Altair® Flux®** ©1983-2025  
**Altair® FluxMotor®** ©2017-2025  
**Altair® GateVision PRO™** ©2002-2025  
**Altair® Geomechanics Director™** ©2011-2022  
**Altair® HyperCrash®** ©2001-2023  
**Altair® HyperGraph®** ©1995-2025  
**Altair® HyperLife®** ©1990-2025  
**Altair® HyperMesh®** ©1990-2025  
**Altair® HyperMesh® CFD** ©1990-2025  
**Altair® HyperMesh® NVH** ©1990-2025  
**Altair® HyperSpice™** ©2017-2025  
**Altair® HyperStudy®** ©1999-2025  
**Altair® HyperView®** ©1999-2025  
**Altair® HyperView Player®** ©2022-2025  
**Altair® HyperWorks®** ©1990-2025  
**Altair® HyperWorks® Design Explorer** ©1990-2025  
**Altair® HyperXtrude®** ©1999-2025  
**Altair® Impact Simulation Director™** ©2010-2022  
**Altair® Inspire™** ©2009-2025  
**Altair® Inspire™ Cast** ©2011-2025  
**Altair® Inspire™ Extrude Metal** ©1996-2025



**Altair® Inspire™ Extrude Polymer** ©1996-2025  
**Altair® Inspire™ Form** ©1998-2025  
**Altair® Inspire™ Mold** ©2009-2025  
**Altair® Inspire™ PolyFoam** ©2009-2025  
**Altair® Inspire™ Print3D** ©2021-2025  
**Altair® Inspire™ Render** ©1993-2025  
**Altair® Inspire™ Studio** ©1993-20245  
**Altair® Material Data Center™** ©2019-2025  
**Altair® Material Modeler™** ©2019-2025  
**Altair® Model Mesher Director™** ©2010-2025  
**Altair® MotionSolve®** ©2002-2025  
**Altair® MotionView®** ©1993-2025  
**Altair® Multi-Disciplinary Optimization Director™** ©2012-2025  
**Altair® Multiscale Designer®** ©2011-2025  
**Altair® newFASANT™**©2010-2020  
**Altair® nanoFluidX®** ©2013-2025  
**Altair® NLView™** ©2018-2025  
**Altair® NVH Director™** ©2010-2025  
**Altair® NVH Full Vehicle™** ©2022-2025  
**Altair® NVH Standard™** ©2022-2025  
**Altair® OmniV™**©2015-2025  
**Altair® OptiStruct®** ©1996-2025  
**Altair® PhysicsAI™**©2021-2025  
**Altair® PollEx™** ©2003-2025  
**Altair® PollEx™ for ECAD** ©2003-2025  
**Altair® PSIM™** ©1994-2025  
**Altair® Pulse™** ©2020-2025  
**Altair® Radioss®** ©1986-2025  
**Altair® romAI™** ©2022-2025  
**Altair® RTLvision PRO™** ©2002-2025  
**Altair® S-CALC™** ©1995-2025  
**Altair® S-CONCRETE™** ©1995-2025  
**Altair® S-FRAME®** ©1995-2025

**Altair® S-FOUNDATION™** ©1995-2025  
**Altair® S-LINE™** ©1995-2025  
**Altair® S-PAD™** © 1995-2025  
**Altair® S-STEEL™** ©1995-2025  
**Altair® S-TIMBER™** ©1995-2025  
**Altair® S-VIEW™** ©1995-2025  
**Altair® SEAM®** ©1985-2025  
**Altair® shapeAI™**©2021-2025  
**Altair® signalAI™**©2020-2025  
**Altair® Silicon Debug Tools™**©2018-2025  
**Altair® SimLab®** ©2004-2025  
**Altair® SimLab® ST** ©2019-2025  
**Altair® SimSolid®** ©2015-2025  
**Altair® SpiceVision PRO™** ©2002-2025  
**Altair® Squeak and Rattle Director™** ©2012-2025  
**Altair® StarVision PRO™** ©2002-2025  
**Altair® Structural Office™** ©2022-2025  
**Altair® Sulis™**©2018-2025  
**Altair®Twin Activate®**©1989-2025  
**Altair® UDE™** ©2015-2025  
**Altair® ultraFluidX®** ©2010-2025  
**Altair® Virtual Gauge Director™** ©2012-2025  
**Altair® Virtual Wind Tunnel™** ©2012-2025  
**Altair® Weight Analytics™** ©2013-2022  
**Altair® Weld Certification Director™** ©2014-2025  
**Altair® WinProp™** ©2000-2025  
**Altair® WRAP™** ©1998-2025  
  
**Altair HPCWorks®, a HPC & Cloud Platform**  
**Altair® Allocator™** ©1995-2025  
**Altair® Access™** ©2008-2025  
**Altair® Accelerator™** ©1995-2025  
**Altair® Accelerator™ Plus** ©1995-2025  
**Altair® Breeze™** ©2022-2025

**Altair® Cassini™** ©2015-2025  
**Altair® Control™** ©2008-2025  
**Altair® Desktop Software Usage Analytics™ (DSUA)** ©2022-2025  
**Altair® FlowTracer™** ©1995-2025  
**Altair® Grid Engine®** ©2001, 2011-2025  
**Altair® InsightPro™** ©2023-2025  
**Altair® InsightPro™ for License Analytics** ©2023-2025  
**Altair® Hero™** ©1995-2025  
**Altair® Liquid Scheduling™** ©2023-2025  
**Altair® Mistral™** ©2022-2025  
**Altair® Monitor™** ©1995-2025  
**Altair® NavOps®** ©2022-2025  
**Altair® PBS Professional®** ©1994-2025  
**Altair® PBS Works™** ©2022-2025  
**Altair® Simulation Cloud Suite (SCS)** ©2024-2025  
**Altair® Software Asset Optimization (SAO)** ©2007-2025  
**Altair® Unlimited™** ©2022-2025  
**Altair® Unlimited Data Analytics Appliance™** ©2022-2025  
**Altair® Unlimited Virtual Appliance™** ©2022-2025  
  
**Altair RapidMiner®, a Data Analytics & AI Platform**  
**Altair® AI Hub** ©2023-2025  
**Altair® AI Edge™** ©2023-2025  
**Altair® AI Cloud** ©2022-2025  
**Altair® AI Studio** ©2023-2025  
**Altair® Analytics Workbench™** ©2002-2025  
**Altair® Graph Lakehouse™** ©2013-2025  
**Altair® Graph Studio™** ©2007-2025  
**Altair® Knowledge Hub™** ©2017-2025  
**Altair® Knowledge Studio®** ©1994-2025  
**Altair® Knowledge Studio® for Apache Spark** ©1994-2025  
**Altair® Knowledge Seeker™** ©1994-2025  
**Altair® IoT Studio™** ©2002-2025  
**Altair® Monarch®** ©1996-2025

**Altair® Monarch® Classic ©1996-2025**

**Altair® Monarch® Complete™ ©1996-2025**

**Altair® Monarch® Data Prep Studio ©2015-2025Altair® Monarch Server™ ©1996-2025**

**Altair® Panopticon™ ©2004-2025**

**Altair® Panopticon™ BI ©2011-2025**

**Altair® SLC™ ©2002-2025**

**Altair® SLC Hub™ ©2002-2025**

**Altair® SmartWorks™ ©2002-2025**

**Altair® RapidMiner® ©2001-2025**

**Altair One® ©1994-2025**

**Altair® CoPilot™©2023-2025**

**Altair® Drive™©2023-2025**

**Altair® License Utility™ ©2010-2025**

**Altair® TheaRender® ©2010-2025**

**OpenMatrix™ ©2007-2025**

**OpenPBS® ©1994-2025**

**OpenRadioss™ ©1986-2025**

### **Third Party Software Licenses**

For a complete list of Altair Accelerator Third Party Software Licenses, please click [here](#).

## Technical Support

Altair provides comprehensive software support via web FAQs, tutorials, training classes, telephone and e-mail.

### Altair One Customer Portal

Altair One (<https://altairone.com/>) is Altair's customer portal giving you access to product downloads, Knowledge Base and customer support. We strongly recommend that all users create an Altair One account and use it as their primary means of requesting technical support.

Once your customer portal account is set up, you can directly get to your support page via this link: [www.altair.com/customer-support/](http://www.altair.com/customer-support/).

### Altair Training Classes

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

# Index

## Special Characters

.cshrc [142](#)  
.profile [142](#)

## A

advanced information [119](#)  
avoid expensive reruns [34](#)

## B

basic change propagation [20](#)  
bin/bash [142](#)  
bin/csh [142](#)  
bin/tcsh [142](#)  
browser interface [85](#)

## C

check all files in a project [35](#)  
clean directories [55](#)  
clean out stuck, unproductive nodes [108](#)  
client/server architecture [10](#)  
command line interface [95](#)  
compute cones with vovcone [18](#)  
concepts and terminology [10](#)  
connect to a project [131](#)  
consumable resources [137](#)  
create a FlowTracer project [126](#)  
create a new set [64](#)  
create sets with selection rules [113](#), [113](#)  
customize actions needed to enable access to Altair Accelerator products on Windows [146](#)  
customize the GUI [67](#)  
customize the web user interface [92](#)

## D

dequeue and stop jobs [38](#)  
destroy a project [134](#)

## E

each project needs a server host [127](#)  
enable Altair Accelerator for non-interactive shells [144](#)  
enable CLI access on UNIX [142](#)  
enable CLI access on Windows [145](#)  
enable the command prompt to communicate with a running product server [147](#)

enable the shell to communicate with a running product server [144](#)  
examples of up-cone and down-cone [16](#)

## F

files, jobs, nodes and sets [12](#)  
find files and jobs [45](#)  
find, remove and clean files [43](#)  
FlowTracer for UNIX users [148](#)  
FlowTracer roles [7](#)  
FlowTracer user guide intro [5](#)  
forget a set [66](#)  
forget nodes [58](#)

## G

graphical taskers LED monitor [66](#)  
graphical user interface [62](#)  
GUI console [62](#)

## I

icons [70](#)  
impact of changing a file or node [105](#)  
incompatible fields [113](#)  
inputs and outputs [15](#)  
invalidate dependent nodes [53](#)  
invalidation and rerunning [20](#)

## J

job life cycle during a run [32](#)  
job queue [36](#)  
job resources [137](#)

## K

keyboard shortcuts [77](#)  
kill processes using the CLI [41](#)

## L

list files in directory showing project status [97](#)  
list inputs and outputs of a node for navigation [107](#)  
list jobs from project having a given directory [99](#)

## M

method 1: use windows explorer to set command line environment [145](#)  
method 2: using Windows command prompt to set command line environment [146](#)

miscellaneous [142](#)  
monitor taskers [83](#)

## N

navigate the flow graph [82](#)  
negated resources [137](#)  
node selection [79](#)  
node status [23](#)

## P

predicates, in selection rules [113](#)  
priority [30](#)  
project creation command [127](#)  
project creation overview [126](#)  
project home page [86](#)

## Q

quantitative resources [137](#)  
query the vovserver [46](#)

## R

regular expressions in selection rules [117](#)  
relationship between FlowTracer, Accelerator, and Monitor [12](#)  
remove files [59](#)  
remove large files that are easy to recreate [60](#)  
remove nodes from the graph [56](#)  
rename nodes [140](#)  
reports [156](#)  
rerun failed jobs [32](#)  
rerun from browser [26](#)  
rerun from GUI [27](#)  
rerun from the command line interface [28](#)  
rerun modes [29](#)  
rerun the down cone of a file [29](#)  
resource management [137](#)  
resource monitoring [139](#)  
run and rerun [26](#), [26](#)  
run Tcl scripts that use Tcl extensions [109](#)  
run vovtsd as a different user [122](#)  
run, rerun, stop, kill jobs [26](#)

## S

sanity check for vovserver [134](#)  
search log files [111](#)



- select a set [64](#)
- selection rules [113](#)
- set browser [63](#)
- set status [21](#)
- set viewer [66](#)
- set, create new [64](#)
- set, forget [66](#)
- set, select [64](#)
- sets [20](#)
- sets of notes [20](#)
- show a list of jobs [43](#)
- show information about the current project [102](#)
- show the history of a file or job within the project [103](#)
- shutting down a project [132](#)
- start a remote vovtasker with vovtsd [122](#)
- start run - run jobs depending on changed inputs [100](#)
- start the vovserver [130](#)
- status of a set [21](#)
- stop the server [132](#)
- system log [67](#)

## T

- taskers: vtk\_tasker\_set\_timeleft [119](#)
- time specifications [117](#)
- troubleshoot taskers [120](#)
- troubleshooting [124](#)
- troubleshooting the retrace [124](#)
- troubleshooting the UNIX setup [145](#)
- troubleshooting: cannot enable project [132](#)

## U

- uncountable resources [137](#)
- up-cones and down-cones [15](#)
- use vovselect for querying [49](#)
- using the set browser and set viewer [71](#)

## V

- verify access to Altair Accelerator products [143](#)
- verify context is working [147](#)
- vls [98](#)
- VOV [117](#)
- VOV introduction [9](#)
- VOV projects [12](#)
- VOVARCH [10](#)
- vovbuild [70](#)

vovcleandir [56](#)  
VovDate [117](#)  
vovinvalidate [54](#)  
vovkill [41](#)  
VovParseTimeSpec [117](#)  
vovproject [130](#), [134](#)  
vovps [42](#), [110](#)  
vovremove [60](#)  
vovresourced [134](#)  
VovScanClock [117](#)  
vovsearchlog [111](#)  
vovselect [46](#)  
vovserver [131](#), [134](#)  
vovset [21](#)  
vovsh [110](#)  
vovshow [43](#)  
vovtaskermgr [134](#)  
vovtouch [53](#)  
vovtsd [121](#)  
vovtsd (taskers service daemon [121](#)  
vovtsd on UNIX [123](#)  
vsc [105](#)  
vsh [104](#)  
vsi [102](#)  
vsm [83](#)  
vsr [100](#)  
vsx [107](#)  
vsy [106](#)  
vsz [108](#)  
vtk\_resourcemap\_set [30](#)  
vtk\_time\_pp [117](#)

## W

weight driven placement [79](#)  
why a node has its current status [106](#)