WorkloadXelerator™ User Guide

Version: 2017.12

Runtime Design Automation
an Altair Engineering Inc. company
www.rtda.com
info@rtda.com

Table of Contents

Table of Contents

1 Introduction

1.1 WX: System Overview

WorkloadXelerator is a high-performance hierarchical scheduler designed for distributed High Performance Computing (HPC) environments. It is based on the patented concepts described in US Patent 9,658,893 about multi-layered resource scheduling.

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

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

1.1.1 Theory of Operation

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

Clients consist of *regular* clients that request a particular service, *slaves* (server farms) that provide computing resources, and *notify* clients that listen for events.

A fresh instance of WX typically has only one persistent or permanent "slave", dedicated to launching requests to get more slaves from the underlying base scheduler, depending on the workload.

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

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

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

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

1.1.2 Examples of Modes of Operation

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

1.1.2.1 Single User Mode, Persistent

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

1.1.2.2 Single User Mode, On-Demand

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

1.1.2.3 Multi User Mode, Persistent

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

1.1.3 What Is New in 2017

The Tcl daemon `vovelasticd` is now replaced by a binary daemon `vovwxd`, which provides support for multi-user WX with a single daemon.

WX currently supports NC, with PBS in the plan for the future.

For each bucket of jobs in WX, the resource expression is internally augmented to be "Bucket:BUCKETID OR (*RESOURCE_EXPRESSION*)", to allow precise routing of jobs from a bucket to the slaves that have been requested for that bucket. This is not visible to the user.

1.2 Access to More Documentation

1.2.1 Documentation Formats: HTML - PDF - Text

All reference documentation is available in three formats: HTML, PDF, and TXT. The main directory holding the documentation is `$VOVDIR/doc`.

1.2.2 Using the Browser to View HTML and PDF Documentation

When the program VovServer is running, you can use any modern browser to visit its built-in web application and the Runtime Products bookshelf of documentation. All you need to know is the URL to the web application.

You can get the URL by running the Command Line Interface program `vovbrowser` to ask the system for it. It will tell you the basic HTTP address and the port number to use.

Example of getting the URL to the web application

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

Copy and paste the URL output from `vovbrowser` into your browser. This will bring up the main page of the web application. The link to the bookshelf page is on the main page and is accessed by clicking on the bookshelf icon. When you get to the bookshelf page, you can see the list of documents and can choose to view the documents in HTML format or as PDF files.

You can also create a direct link to the bookshelf URL based on what `vovbrowser` reports. Change the trailing token "project" to "cgi/booskhelf.cgi".

Example of converting the above URL to a direct link to documentation

```
http://comet:6271/cgi/bookshelf.cgi
```

Note: To view all available cgi scripts, features available via GUI, go to `http://`*`host`*`:`*`port`*`/cgi/listcgi.cgi`.

1.2.3 Searching Documents

There is a search field on the Documentation Bookshelf that allows you to perform searches for words within the manuals.

Enter your search criteria and click on the Search button and you will get a search result list containing the name of each html page that contains the search criteria, along with the line from that file containing the search criteria. The file name is a link to that page so you can navigate there to read the page.

The search field is at the top of the browser window for every page you view when you read the manuals as html.

1.2.4 Using the Browser to View Local HTML Documentation Files

The static HTML files of the document bookshelf are located in `$VOVDIR/doc/html`. These can be viewed in the browser as local HTML files without using a web server. This is the way to see the documentation when the VovServer program is not running.

You can enter the URL to the bookshelf's main HTML file using the "file:" protocol instead of the "http:" protocol.

- `file:`*`/wherever/vov/is/installed/common`*`/doc/html/bookshelf.html`

1.2.5 Plain Text Documentation

The documentation also exists as plain text files, in case you prefer to read them in this format.

The text files are in `$VOVDIR/doc/txt`. Each file is a conversion of the corresponding HTML page, as formatted by the public domain browser "lynx", with a file name having a suffix of ".txt".

1.2.6 Command Line Interface to View Text Documentation

You can run the command `vovdoc` to search the plain text documentation. Here is the usage message for this command:

Usage: `'vovdoc'`

```
vovdoc: Usage Message

  DESCRIPTION:
      The utility vovdoc scans the files in $VOVDIR/doc/txt/*/*
      for documentation that matches the words provided on
      the command line.

  SYNOPSIS:
      % vovdoc [OPTIONS] word ...

  OPTIONS:
     -h           -- This help
     -v           -- Increase verbosity
     -n  <N>      -- How many documents to show (default 1)
     -s           -- Show keywords with <<<WORD>>>
     -test        -- Test installation of txt docs.

  EXAMPLES:
     % vovdoc vovbuild
     % vovdoc vovid
     % vovdoc vovid -s
     % vovdoc exit status
```

1.2.7 More Ways to Access Documentation

Another source of live information is using the command `vovshow`, which is described in detail in `vovshow`. The following options are often useful:

- `vovshow -env RX` shows the environment variables that match the regular expression RX provided.
- `vovshow -fields` shows the fields known to the version of VOV in use.

- `vovshow -failcodes` shows the table of known failure codes.

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

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

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

1.3 WorkloadXelerator Brief Help

WorkloadXelerator has two main commands:

- **`wx`** is used to submit, query, and stop jobs. This command can also be invoked as **vnc** or as **nc**.
- **`wxmgr`** is used to start and stop a "queue", i.e. a WorkloadXelerator instance.

Here are the usage messages generated by these commands.

Usage: `'wx'`

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

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

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

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

  Unique abbreviations for commands are accepted.

  Advanced features:
    cmd <command>       Execute an arbitrary VOV command in the
                        context of the WorkloadXelerator server.
    source <file.tcl>   Source the given Tcl file.
    -                   Accept commands from stdin.

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

  Copyright (c) 1998-2018, Runtime Design Automation.
```

Usage: `'wxmgr'`

```
wxmgr: Usage Message

         This program manages the vovserver for WorkloadXelerator.
         Copyright (c) 1998-2018, Runtime Design Automation.

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

  ACTIONS:
         info   [-queue|-q <name>] [-v]
         reset  [-soft | -hard | -h ]
         start  [-dir <server_working_dir>] [-force] [-queue|-q <name>]
                [-port <port> ] [-webport <port>] [-roport <port>]
                [-dbhost <host>] [-dbroot <path>] [-prod nc|wx]
                [-basequeue <name>]
                The default <server_working_dir> is
                /remote/zfbuild/integ/builds/201712u3/wx.
                This is the parent of the configuration (.swd) directory for
                the queue.
         stop   [-force] [-freeze] [-queue|-q <name>]
                -force   Do not prompt for confirmation
                -freeze  Instruct slaves to keep running and wait for a new server
         cm     [-queue|-q name] <ACTION> [ARGUMENTS]
                Configuration Management. Pass "help" for detailed usage.

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

  EXAMPLE TO STOP AND RESTART SERVER:

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

1.4 Command Line Interface

All user commands have the following structure:

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

The *subcommand* is one of the following: clean cmd debug dispatch forget getfield gui help hosts info list jobclass modify monitor rerun resume run source stop summary suspend wait

For example:

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

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

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

2 Getting Started

2.1 Setting Up the User Shell

To set up your shell to use WorkloadXelerator, you need to know where the software has been installed. Ask your system administrator.

2.1.1 User setup: C-Shell, TCSH

You have two choices:

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

```
# Add this to your .cshrc
source /wherever/vov/is/installed/common/etc/vovrc.csh
```

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

```
% cd /wherever/vov/is/installed
% cd common/scripts
% ./vovsetupuser -csh
```

2.1.2 User setup: Bourne Shell, K-Shell, Z-Shell, BASH

You have the same two choices, except that the file to be sourced is called `.vovrc.sh`.

- Source the file `$VOVDIR/etc/vovrc.sh`. We recommend that you add the following line to your `~/.profile` file:

```
# Add this to your .profile or .bashrc
. /wherever/vov/is/installed/common/etc/vovrc.sh
```

- Run the `vovsetupuser` script.

```
% cd /wherever/vov/is/installed
% cd common/scripts
% ./vovsetupuser -sh
```

2.1.3 Verify the Setup

You can verify the setup by running simple commands.

The command `vovarch` shows the operating system of the machine you are working on. Example:

```
% vovarch
linux64
```

The command `vovversion` shows the version of WorkloadXelerator that is installed. Example:

```
% vovversion
2017
```

3 What Happens While Jobs Are Running

3.1 What to Look For When Submitting Jobs

Note: To submit a job, the working environment must be set. For information, refer to the section Set up UNIX Shell to Access NetworkComputer

3.1.1 Information Displayed From a Submitted Job

By default, `wx run` provides information when a job is submitted. The following example shows information that is output with a simple command. The amount of information displayed is determined by the verbose level. In the following example, verbose is at the default level of 4.
Note: The environment is set with a *snapshot*.

Example:

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

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

The amount of information returned at submission time can be changed by setting the verbose level with option `-v LEVEL`, as in

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

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

3.2 Scheduled jobs

Jobs that cannot be dispatched immediately due to a delay in acquiring the resources from the underlying scheduler. such as CPUs or software licenses, are put on the job queue. If the base scheduler is NC, then the fairshare policy in that queue primarily determines what resources (slaves) are made available to WX and those resources are allocated to WX jobs of a specific FairShare group. This effectively by-passes the WX FairShare algorithm. The rules described below are applicable to other base schedulers:

- Scheduling is first determined by the FairShare mechanism. All active FairShare groups, all groups with queued jobs, are ranked based on their distance from the target share of computing resources and the current number of running jobs. The FairShare group that is farthest behind the target has rank 0 (zero) and is selected first for scheduling. If none of the jobs from the FairShare group with rank 0 can be dispatched, WorkloadXelertor looks at the jobs for the FairShare group for rank +1 and so one.
- For a given FairShare group, jobs with higher priority are scheduled ahead of lower priority jobs.
- For a given FairShare group of a given priority, jobs are scheduled on a first-come first-serve basis.

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

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

4 Submitting Jobs to WX

4.1 Queue Selection

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

If you want to use the default WorkloadXelerator queue (i.e. "wx"), you can say:

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

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

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

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

Examples:

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

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

Example:

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

4.2 Submitting Jobs with CLI Commands

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

4.2.1 Example: Submit a Single Job

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

When submitting a single job with no options:

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

Examples:

- **Running a job, no specifications:**

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

- **Running a job, environment specified (-e)**

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

- **Running a job, with environment and resources specified (-r)**

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

- **Running a job, with environment and resources specified, and limited verbosity (-v)**

```
# Control verbosity: print the jobId only.
% wx run -v 1 -e BASE -r RAM/20 -- sleep 30
00002579
# Running a job, environment and resources specified, limited verbosity and wait for job to finish (-w):
% wx run -w -v 0 -e BASE -r unix -- sleep 30
```

4.2.2 Example: Submit Multiple Jobs

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

Note: When submitting multiple jobs, the same environment, the same resources, and the same priority level are used for each job. Each job has its unique VovId.

1. Prepare a file with one command on each line. (Empty lines and comments are ignored.)

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

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

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

4.2.3 Default output of 'wx run '

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

- The resource list assigned to the job, which can be controlled with the option `-r`.
- The environment used for the job, which can be controlled with the option `-e`.
- The command line.
- The log file used to store both `stderr` and `stdout` of the command, which can be controlled with the option `-l`
- The JobId assigned by WorkloadXelerator to this job. href="vovid.html">JobIDs are used as handles with many of the WorkloadXelerator commands.

4.3 Job Arrays

Using an array provides the option of submitting multiple jobs in a specific order. Each job in an array is assigned its own job ID and is treated as an individual job. The syntax is: `wx run -array <n>`

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

Example:

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

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

Examples:

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

4.4 Controlling the Environment

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

1. Use a **snapshot of the environment** used at submission time.
   This method is the simplest and is automatically selected if the environment variable `VOV_ENV` is not defined. The disadvantage of this method is that the snapshot may not be portable across platforms.
   Note: this method is **not available for Windows**.
2. Use a **named environment**, which allows the slave to create the environment on the fly using the VOV Environment Utilities.
   This method offers several advantages: strict control on the environment, greater efficiency, less disk space utilization, easier execution across multiple platforms. This method is used if the environment variable `VOV_ENV` is defined; the value of the variable indicates the name of the environment to use.
   Note: This method is **required for Windows**.

4.4.1 Using Environment Snapshots

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

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

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

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

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

variables are defined in the file `$VOVDIR/tcl/vtcl/vovenvutils.tcl`
An environment snapshot may be shared by many jobs.

When using a snapshot, the job is submitted with environment `SNAPSHOT(name_of_snapshot_file)`.
Note: This is a *named environment*.

To force the creation of an environment snapshot:

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

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

4.4.2 Using Named Environments

The NetworkComputer Environment Utilities consist of two commands: `vel`, lists the available environments; `ves` switches between environments. For more information, refer to the Environment Management chapter of the Flowtracer Administrator Manual.

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

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

4.4.3 Selecting a Named Environment

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

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

4.4.4 Using Snapshot with Named Environment

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

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

4.5 Pre-Command and Post-Command Job Conditions

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

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

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

4.6 Pre-Condition

A pre-condition is executed before the job is run. A pre-condition is executed with the same credentials as the job (userid, os-groupid) and is in the same directory of the job.

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

4.7 Post-condition

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

- When the post-condition script is invoked, the job is still *running*.
- The post-condition is executed after the job, even if the job fails, but it is not executed if the pre-condition fails.
- The exit status of the post-condition becomes the exit status of the job, i.e. you can use the post-condition to affect the exit status

4.8 Submission of jobs with pre-condition or post-condition

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

```
% wx run -pre $VOVDIR/etc/pre/pre_check.csh sleep 10
% wx run -post $VOVDIR/etc/post/post_cleanup.csh sleep 10
```

4.8.1 Log Files

The standard output from the pre- and post-commands is saved in log files in the run directory; standard error is discarded. These files are created with zero size if the pre- and post-commands redirect all the output of the files. At the end of the job, if these files are zero length, they are automatically deleted to reduce disk space overhead.

The log files are named according to the following rules:

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

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

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

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

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

4.9 Using JobClasses

A jobclass allows multiple job parameters to be set in a single object that can be requested at submission time. For example, there may be a job that requires 3 different licenses, 4GB of RAM, and 4 cores. Instead of requesting all 3 licenses, a job class can be created that is called with the `-C` submission option to the `wx run` command. Job classes are often used to emulate *queues* that are found in other batch processing systems.

Note: A jobclass can only be created by a WorkloadXelerator administrator.

4.9.1 Finding the Available JobClasses

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

Example:

Usage: `'wx jobclass -h'`

```
wx: Usage Message

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

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

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

In addition, WorkloadXelerator provides the CGI script "/cgi/jobclasses.cgi". This page shows a table of the job classes, with clickable links to the definitions of each class, and to the sets containing the jobs in that class. It also shows the pass/fail status as a bar graph.

4.9.2 Submitting Jobs Using Job Classes

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

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

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

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

Example:

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

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

4.10 Choosing the FairShare Group

While WorkloadXelerator supports the use of FairShare groups, their utility depends upon the underlying batch system. If the underlying batch system is Network Computer then the recommendation is that no additional fairshare configuration (with access control lists, weights and time windows) is done in the WX queue. Instead WX just passes on the requested fairshare group on to NC where the existing policies and allocations are enforced. For base schedulers other than Network Computer, the WX FairShare model may be used to prioritize jobs of different categories within the WX queue.

In WorkloadXelerator, FairShare groups are managed by either the information in the `vwx.swd` directory that contains the `policy.tcl` file, or the `vovfsgroup` utility. Every user has a default FairShare group, which is set in the `policy.tcl` file. Use `wx run` with the option `-g` to select a different group.

Examples:

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

4.10.1 FairShare Subgroup

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

Examples are shown below:

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

4.11 Prepending and Appending Arguments in a Job Submission

Job submission is also affected by the value of the optional variables `NC_RUN_ARGS` and `NC_RUN_ARGS_AFTER`, which specify a list of arguments that are prepended and appended to the argument list passed to the submission command.

For example, if we define the variables as follows:

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

then the submission

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

becomes effectively

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

4.12 Priority

Two types of priorities are supported:

- **Scheduling priority:** affects 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 slave. The range is 1(low) to 15(top).

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

- As the use of fairshare weightings are not currently supported in WorkloadXelerator, for the jobs of a given user, higher priority jobs are scheduled before lower priority ones.
- A low priority job will be dispatched before a high priority job if the resources for the low priority job are available while the resources for the high priority job are not.

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

Example:

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

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

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

4.12.1 Priorities Relative to Previous Run

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

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

Example: rerun the job 123456 with increased priority:

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

5 Modifying Running Jobs

5.1 Running Interactive Jobs

Interactive jobs require attention as they run, whereas batch jobs are run unattended. Interactive jobs are only supported on Unix in NetworkComputer. There are three types of interactive jobs, which are described in the table below:

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

5.1.1 When to Use -Il and -Ir

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

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

5.1.2 Capturing the Log for an Interactive Job

NetworkComputer does not use a wrapper (such as vw) when running an interactive job; it is more complex to capture a log of all I/O. The recommended method to capture the log is to use the Unix utility script as shown th example below:

```
% wx run -Ir .... script ./my_log_file -c "gdb /bin/ls"
```

5.1.3 Jobs that Require a DISPLAY: option -Ix

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

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

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

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

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

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

5.2 Modifying Scheduled Jobs

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

Usage: 'wx modify'

```
wx: Usage Message

   WX MODIFY:
          Modify scheduled jobs

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

   OPTIONS:
          -h                      -- This help.
          -v                      -- Increase verbosity.
          -showfields             -- Show fields that can be modified.
          -set         <SETNAME>  -- Modify all jobs in the specified set.
          -<FIELDNAME> <NEWVALUE> -- Set the specified field to the specified value.

   EXAMPLES:
          % wx modify -jobclass short 0012345
          % wx modify -res License:xxx 0012345
          % wx modify -jobname superman 0012345
          % wx modify -res License:xxx -set MySet
          % wx modify -group /time/users !
          % wx modify -jpp smallest -numa pack 0012345
```

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

Usage: 'wx modify -showfields'

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

Examples:

```
% wx modify -showfields
% wx modify -autokill 30m 0012345
```

5.2.1 Restrictions and consequences

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

- **autokill** (duration job runs before being killed automatically)
- **fstokens** (fairshare tokens; when changed, moves the job to the proper fairshare bucket)
- **xpriority** (execution priority)
- **jobname** (can only be changed by job's owner)
- **jpp** (job placement policy)
- **legalexit** (if the job is not running, the exit status is checked again with the new legal values; if it does not match, the job is invalidated)
- **numa** (job placement non-uniform memory access policy; linux only)
- **preemptable** (flag indicating the job can be preempted)
- **priority** (schedule priority)
- **systemjob** (flag that indicates it is a system job; not normally modified this way)
- **xdur** (expected duration)

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

- **autoflow** (flag that indicates a job should be skipped; if changed, job is moved to proper bucket)
- **nojournal** (flag that turns off journal entries for the job)
- **res** (resources; if changed for a scheduled job, the job is re-queued)
- **scheddate** (date/time job was scheduled)

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

- **autoforget** (flag that indicates the job will be forgotten after completion)
- **cmd** (command line of job; job is invalidated if changed)
- **deadline** (the desired date/time the job should be completed)
- **dir** (filesystem path where job runs; job is invalidated if changed)
- **env** (named environment of job; job is invalidated if changed)
- **group** (fairshare group; if changed for a scheduled job, the job is re-queued)
- **jobclass** (resource group; if changed for a scheduled job, the job is re-queued)
- **jobproj** (project associated with job; if changed for a scheduled job, the job is re-queued)
- **res,aux** (aux resources; if changed for a scheduled job, the job is re-queued)
- **submithost** (host from which job is submitted)
- **tool** (name of tool associated with command)

6 Monitoring Workload

6.1 Getting Information About a Job

The command `wx info` displays information about a job.

Usage: 'wx info'

```
wx: Usage Message
  WX INFO:
      Get information about a specific job or list of jobs.
  USAGE:
      % wx info [options] <jobId> ...
  OPTIONS:
      -h               -- Show this message.
      -v               -- Increase verbosity.
      -l               -- Show the log file (actually, it shows all outputs).
      -e               -- Show the environment name, or contents if a snapshot.
      -c               -- Show slave compatibility table (which slave can run a job).
      -sc              -- Show slave compatibility in normal output.
      -dep             -- Show job dependencies.
      -J   <jobname>   -- Show the jobs with given name.

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

When executed with just a jobId, the command displays information such as start time, completion time, duration, resources, priority, exit status, as shown here:

```
% wx info 001241618
Id,User,Group   001241618,jon.staff,/time/users.jon
Environment     BASE
Directory       /Users/jon/work
Command         hostname > wx_logs/20180404/115829.25846
Resources       macosx CORES/1 RAM/500
Submitted from  mac01ac
Submitted at    Wed Apr 04 11:58:29 CEST 2018
Priorities      schedule=normal execution=low
PlacementPolicy first,pack
Status          Done
   Host         mac01ac
   Slave        nc_vnc_033524434
   QueueTime    8s
   CPUTime      0.03
   MaxRAM       0MB
   Duration     0s
   Age          1s
   AutoForget   1
Job 001241618 is Done

Main Reason:
   This job successfully executed.

Additional Information:

   WX Information:

   Slave submitted to NC queue: vnc
   Slave job ID in NC queue:    033524434
```

The option `-l` can be use to look at the log file for the job

```
% wx info -l 001241618

Log file is: db=FILEX '/Users/jon/work/wx_logs/20180404/115829.25846'
mac01ac.local
```

6.2 Getting detailed information about a job

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

Usage: 'wx getfield'

```
wx: Usage Message

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

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

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

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

Examples:

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

6.3 Status of Jobs

In WorkloadXelerator, each job goes through a number of states, described in the following table:

Status	Color	Description
Idle	BlueViolet	If the node is a job, either it has not been run successfully yet or it needs to be run again, because one of its inputs has been modified since the last time the job was executed. If the node is a file, it is the output of a job that is not Idle.
Queued	Light blue	The job is scheduled to be run. It may be already queued or it will go in the queue as soon as all its inputs are ready.
Running	Orange	The job is currently being retraced; it has been dispatched to one of the slaves. All the outputs of such a job are either RETRACING or RUNNING.
Done	Green	If the node is a job, it has run successfully. If the node is a file, it is up-to-date with respect to all other files and jobs on which it depends.
Failed	Red	The job ran and failed.
Transfer	Cream	The job is being transfered to another cluster and it is not yet running.
Suspended	Pink	The job was running (or retracing) and one of the processes belonging to the job is currently suspended.
Sleeping		Either the job caused an output conflict upon submission (bad dependencies) or the job was not reclaimed by any slave upon crash recovery.
Withdrawn	Gray	

		A job has been withdrawn after dispatching, such as by the preemption daemon. **Note:** This status occurs rarely and tends to be hard to observe.

The normal sequence for a successful job is:

Idle Queued Running Done

The normal sequence for a failing job is:

Idle Queued Running Failed

6.4 Job Runtime Monitoring and Profiling

When a job is running through a vovslave, the slave automatically monitors RAM and CPU utilization of the job, including all of its children. Job statistics are sampled about once a minute. This data sampling rate does not capture jobs that complete in less time than the than the sampling period.

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

6.5 CPU Progress and Run Status Indicators

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

- **CPUTIME** : The total accumulated CPU time in milliseconds.
- **CPUPROGRESS**: Percentage of CPU accumulated in the unit time. For example, if in 60 seconds a job uses 60 seconds of CPU time, then the CPUPROGRESS is going to be 100. This field can be 0 (zero) for jobs that are stuck: *holding onto* the CPU resource but not running, which makes the CPU unavailable for other jobs. This field can also be greater than 100 for multithreaded jobs.

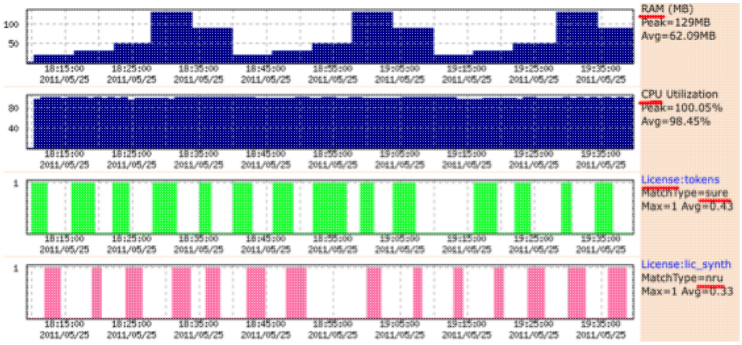
- **LASTCPUPROGRESS**: A timestamp indicating the last time CPU usage has increased. This is used to to identify stuck jobs.
- **RUNSTATUS**: A descriptive text field that shows how well the job is doing. Some typical values are Good, Paging, NoCpu. The complete list of values is shown below.

6.5.1 Values of the RUNSTATUS Field

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

6.6 Job Profiling

When *job profiling* is activated, WorkloadXelerator tracks memory usage, CPU usage and license usage over the lifetime of a job. The output of job profiling is a set of plots as shown below:



The first plot from the top shows the RAM usage over time. The second plot shows CPU usage over time. The third plot shows usage for licenses that were requested at submission time. The fourth plot show usage of a license that was not requested ("Requested/Used")

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

```
% wx run -profile myJob
```

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

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

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

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

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

6.7 Listing Jobs

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

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

The WorkloadXelerator Administrator may have configured methods to mitigate this load, including caching and use of a subserver process to handle `wx list` commands.

The command `wx list` supports the following options:

Usage: `'wx list -h'`

```
wx: Usage Message

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

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

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

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

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

  USAGE:
        % wx list [OPTIONS]

  OPTIONS:
        -O <format>       -- Specify a different Output format.
                             Refer to the manual for a description
                             of formats. For experts.
                             This can also be specified via the NC_LIST_FORMAT
                             environment variable.
        -O+ <format>      -- Add one or more fields to the default output format.
        -H                -- When used with -O, show header line.
        -l                -- Long format: show group, user, host,
                             and full command.
        -L                -- Very long format: show start and end dates, duration.
        -c                -- Count jobs: this option affects the output format.
                             It adds a column with the the position of each
                             job in the list of jobs to be shown.
        -S <rule>
        -select <rule>
        -selrule <rule>   -- 3 ways to specify the selection rule. For experts.
                             The clause 'isjob' is added to the selection rule.
        -S+ <rule>        -- Alternate (OR) selrule; jobs that match any
                             selrule will be shown.
                             The clause 'isjob' is added to the selection rule.

        -subjobs          -- Show also sub-jobs too (e.g. jobresumer, partialtool)
        -systemjobs       -- Show also system-jobs
        -alljobs          -- Show all types of jobs,
                             including sub-jobs and system-jobs.
        -a                -- Show jobs for all users.
        -r                -- Show only running jobs.
        -s                -- Show only suspended jobs.
        -q                -- Show only queued jobs.
        -u <user>         -- Show jobs belonging to given user. Ignored if
                             used with -selrule or -set option.
        -first <index>    -- Show jobs starting at index. By default, first index is 1.
        -last  <index>    -- Show jobs ending at index. By default, last index is -1, which is the last job.
        <num>             -- Show first <num> jobs if <num> is positive. Show last <num> jobs if <num> is negative number
        -J <jobName>      -- Show only jobs with given job name.
                             WARNING: this option can place a high load on the WorkloadXelerator server
                             in large workload environments due to the need to compare
                             strings for each job, it is recommended to avoid calling
                             this unless truly required.
        -set <setName>    -- Show only jobs in the given set.
                             This option can be repeated to show the content
                             of multiple sets.  If a job belongs to multiple
                             sets, it may be reported more than once.
        -dir <directory>  -- Show only jobs in the given directory.
        -ss               -- Use subserver, if available.
        -cache            -- Use result cache. It is recommended to utilize the result
                             cache to reduce the load on the WorkloadXelerator server in cases
                             where list queries are scripted. By default, the cache expires
                             after 30s.
        -noss             -- Do not use subserver, even if available. (Default)
        -v                -- Increase verbosity.
        -h                -- Print brief usage.
```

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

Option	Description
-a	List jobs of all users.
-dir dir_name	List only the jobs in the given directory.
-h	Print the help message for list.
-l	Use a long format to show user and host in addition to the JobID, status and command.
-L	Use an even longer format, to show start and finish time.
-n N	List the last N jobs instead of the default 20.
-selrule <rule>	List jobs that match the given selection rule.
-O format_string	Specify output format. The default format is `"@ID@ @STATUS:10@ @COMMAND@"`.
-u username	List jobs that belong to the specified user.

6.8 Summary of All Jobs

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

Usage: `'wx summary -h'`

```
wx: Usage Message

  WX SUMMARY:
        Get a summary report for all of my jobs.
  USAGE:
        % wx summary [options]
  OPTIONS:
        -h                    -- This message.
        -w                    -- Show detailed info about wait reasons.
        -b                    -- Show buckets.
        -P                    -- Print report for all projects
        -p PROJECT            -- Print report for given project (repeatable)
        -u USER               -- Print report for given user (repeatable)
        -all_sets             -- Show all sets
        -a,  -all             -- Print report for all users.
        -all_users            -- Same as -a
        -set SETNAME          -- Show report for just that set.
```

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

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

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

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

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

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

6.9 Invoking the GUI

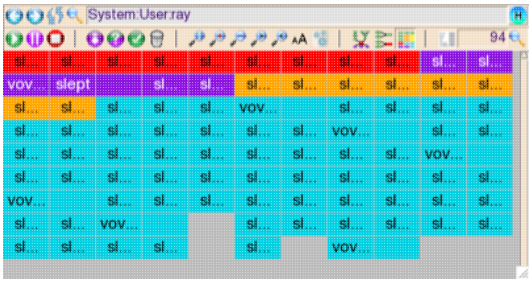
Job execution can be monitored with `wx gui`. This command opens a monitoring tool; no interactive capabilities (such as configuration or running jobs) are provided. Interactive capabilities are available with `wx cmd vovconsole`.

Usage: `'wx gui -h'`

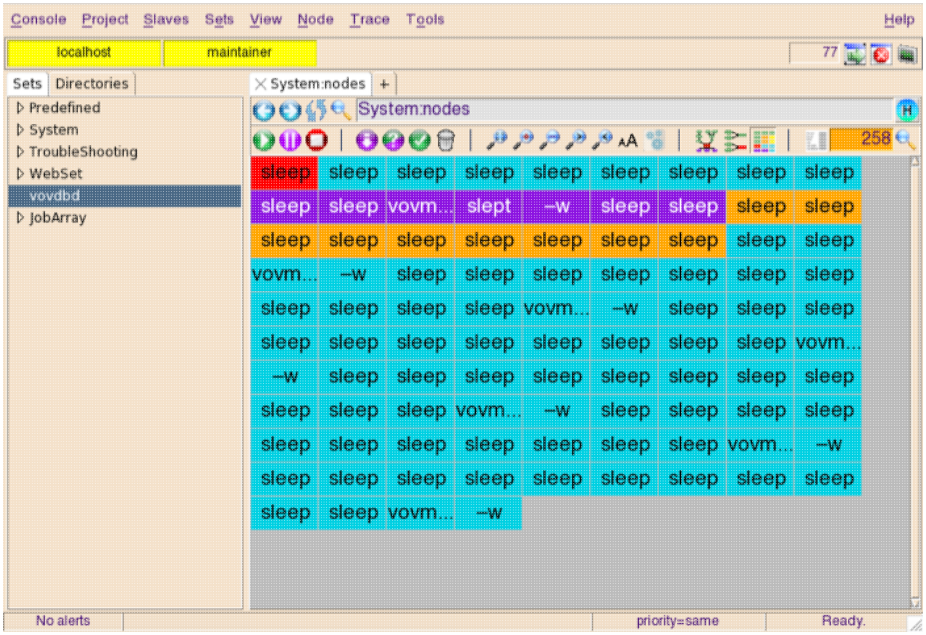
```
wx: Usage Message

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

         -all
         -a                  -- Show all jobs.
         -u <user>           -- Show jobs for specified user.
         -s <SETNAME>
         -set <SETNAME>
         -setname <SETNAME>  -- Show specified set.
         -timeout <TIMESPEC> -- Stop async update after this time (default 2h).

         -ss                 -- Use subserver, if available.
         -noss               -- Do not user subserver, even if available. (Default)
         -submit             -- Activate job submission dialog
         -limitGui <N>       -- Override the limit of 3 max GUI per user.

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

         -metrics            -- Show scheduler metrics.
         -metricsConfig <file> -- Use specified metrics configuration file.
         -slaves             -- Show compact slaves monitor.

         -fontsize <size>    -- Specify the normal font size. Default is 10.
                                Legal range is 3 to 36.

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

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

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

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

Option	Explanation
-all | -a	Show all jobs in the server database. By default, show only my jobs.
-s | -set | -setname <NAME>	Show the jobs in the named set.
-legend	Show legend for the color coding.

```
% wx gui &
```




```
% wx cmd vovconsole &
```



6.10 Icons

All those icons provide descriptions that you can find by hovering on top of the icon.


Run / Retrace / Start: 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.


Dequeue: Un-schedule a job that is not yet running or abort a retrace request. Running jobs are un-harmed and keep running. Graceful Stop. Cyan/asked-to-run jobs will turn back purple/invalid.


Stop: 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: This purple icon shaped like an 'i' merged with a down arrow invalidates the selected nodes.

 Try Validate: tries to validate the selected jobs and its downcone.

 Force Validate: equivalent of make -t.

 Forget: Removes the job/file from the graph of dependencies. This does NOT remove the file from disk. It just removes it from the dependencies allowing blocked jobs to start. Usually rerunning 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.

 Display the current set using the "graph" widget.

 Display the current set using the "horizontal" widget.

 Display the current set using the "grid" widget.

 Stats: Display the current set using the "stats" widget.

 Set the various preferences for Weight Driven Placement. This takes you to the **Graph** tab of the Preferences dialog.

6.11 Showing the Hosts/Slaves

The command `wx hosts` shows the list of the hardware resources currently connected to an instance of WorkloadXelerator. These hardware resources are called "slaves" in WorkloadXelerator parlance. An example is shown below:

Usage: `'wx hosts -h'`

```
wx: Usage Message

      WX HOST
          Show hosts in the farm.
          The hosts are also called 'slaves' or 'vovslaves'.

      USAGE:
          % wx hosts [OPTIONS]

      OPTIONS:
          -h         -- This help
          -c         -- Show consumable resources (e.g. RAM and CPUS)
          -f         -- Show list of slave fields
          -r         -- Show resources (legacy: pre-2013.03 format)
          -m         -- Show machine parameters (RAM, CPUfreq,...)
          -a         -- Show all known hosts (used with -m)
          -hw <HW>   -- Show only slaves that match HW constraints.
          -rule <SELRULE> -- Show only slaves that match the given selection
                             rule. Use "vovselect fieldname from slaves" for
                             the complete list of fields that can be used in
                             the rule. Example rules:
                                 "status==READY"
                                 "status!=OVRLD slots>8"
                             Can accept multiple constraints.
          -O <fmt>   -- Specify output format. The format string can contain
                        elements like @FIELDNAME@  or @FIELDNAME:WIDTH@  where
                        a negative width means left-align and a positive width means right-align.
          -RAM       -- Same as -O "@NAME@ RAM/@RAM@ RAMFREE#@RAMFREE@ RAMTOTAL#@RAMTOTAL@"
          -SLOTS     -- Same as -O "@NAME@ SLOTS/@SLOTS@ SLOTSTOTAL#@SLOTSTOTAL@ CORES/@CORES@ CORESTOTAL#@CORESTOTAL@"
          -LOAD      -- Same as -O ...selection of fields about load...
          -INFO      -- Same as -O ...fields about host, arch, model, ...
          -ALL       -- Show all info about each slave (multiline output)

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

      LEGACY COMMANDS:
          % wx hosts -r
```

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

% wx hosts -r
...
% wx hosts -m
...
```

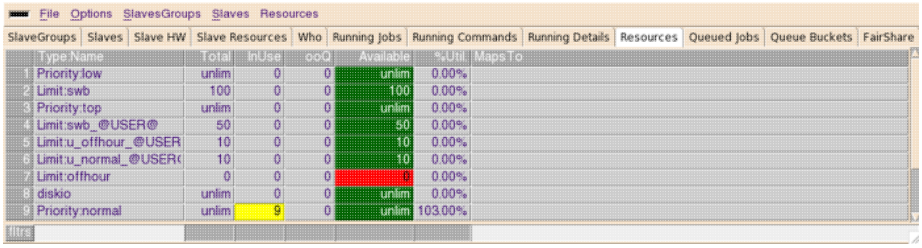
6.12 Monitoring Jobs, Slaves and Resources

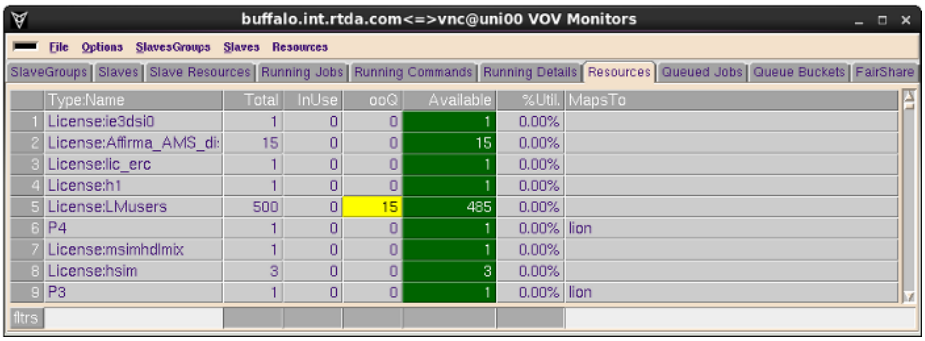
The activity of WorkloadXelerator can be monitored with a dialog as shown below. The dialog is invoked with:

```
% wx monitor
```

1. **SlaveGroups:** The activity of slave groups
2. **Slaves:** The activity of slaves

3. **Slave HW:** The hardware offered by slaves
4. **Slave Resources:** The resources offered by slaves
5. **Who:** Who is running jobs
6. **Running Jobs:** The progress of running jobs
7. **Running Commands:**The details of running commands
8. **Running Details:**The details of running jobs
9. **Resources**The usage and availability of resources
10. **Queued Jobs**The jobs in the job queue
11. **Queue Buckets**The jobs in the job queue organized by groups of similar jobs (called 'buckets');
12. **FairShare:** The FairShare statistics.





7 Managing Jobs - Wait, Stop, Forget

7.1 Waiting, Stopping, Cleaning, Debugging Jobs

To wait for one or more jobs to complete, use the following command:

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

Usage: 'wx wait'

```
wx: Usage Message

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

        -dir <directory>     -- Wait for all jobs in the given directory.
        -subdirs <directory> -- Wait for all jobs in the given directory and subdirectories.
        -select <rule>       -- Wait for all jobs in set defined by 'rule'.
                                The predicate 'isjob' is automatically added
                                to the rule.
        -set <setName>       -- Wait for all jobs in a set.
        -p                   -- Use polling method (automatic delay)
        -poll <MS>           -- Use polling method with a delay of <MS> milliseconds.
                                MS values are between 2000 and 60000.
        -k                   -- Jobs are killed if you interrupt wait by Ctrl-C.
        -l                   -- Show log file of last job while waiting.
                                The log is shown either as it is created
                                if the job has been submitted with option -wl
                                or all at once upon job completion.
        -maxwait <timespec>  -- Specify a maximum wait time.
        -timeout <timespec>  -- Same as -maxwait
        -callback <cmd>      -- Execute 'cmd JOBID JOBSTATUS' for every job
                                that completes. Output to stdout. Errors ignored.
        -file <file>         -- For experts. Source <file>, mostly to define overrides
                                for procedure VncWaitCallback { cmd jobId status }
        -ss                  -- Use subserver, if available.
        -noss                -- Do not use subserver, even if available. (Default)

  RETURN:
        0       -- All jobs are done.
        1       -- Some jobs are still invalid.
        2       -- Some jobs are failed.
        3       -- Some jobs have unexpected status.
        4       -- Some jobs have been lost.
        5       -- Waited for long enough (see -maxwait option)

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

7.1.1 Options

Option	Description
-q	Quiet wait (default)
-v	Verbose wait: meant mostly for debugging, this option prints a selection of the events received while waiting for the jobs to be done.
-dir *dirname*	Wait for all jobs in the specified directory. Only one directory can be specified.

-k	This option is active only in conjunction with the other option `-l`. Its effect is that if the waiting process is terminated by the INT signal (e.g. by a Ctrl-C), then the jobs we are waiting for will also be stopped.
-l	While waiting, show the log file of the last job that we are waiting for.
-select *rule*	Wait for all the jobs that belong to the set defined by the given selection rule. The predicate `isjob` is automatically added to the rule. For example, the rule "`command~spice`" actually becomes "`isjob command~spice`".

7.1.2 Return values when waiting for jobs

Exit status	Meaning
0	All jobs completed successfully.
1	Some jobs could not be executed.
2	Some jobs failed.
3	Some jobs have an unexpected status.
4	Some jobs have been lost.
5	Waited long enough.

7.1.3 Examples:

```
% wx wait -h
% wx wait 22345 22356
% wx wait -dir .
% wx wait -select "tool==spice"
% wx wait -set "spiceset"
```

7.2 Stopping Jobs

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

Usage: 'wx stop'

```
wx: Usage Message

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

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

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

     The default list of signals is also controlled by
     the environment variables VOV_STOP_SIGNALS and NC_STOP_SIGNALS.

     If both NC_STOP_SIGNALS and VOV_STOP_SIGNALS are present in the
     environment, the value of VOV_STOP_SIGNALS will be used.  Their
     functionality is otherwise identical.

     The default list of signals is "NORMAL"

     NORMAL   =  TERM,HUP,INT,KILL

  USAGE:
      % wx stop [OPTIONS] <jobId> ...

  OPTIONS:
      -h                  -- This message
      -set <setname>      -- Stop all my jobs in the given set.
      -J <jobname>        -- Stop all my jobs with given jobname.
      -mine               -- Stop all my jobs.
      -dir <directory>    -- Stop all jobs in the given directory.
      -allusers           -- Stop all jobs (only ADMIN can do it).
      -dequeueonly        -- Just remove jobs from the queue.
                             All currently running jobs are not affected.
                             Can be abbreviated to -d.
      -d                  -- Same as -dequeueonly.
      -signals SIGLIST    -- Comma separated list of signals to send to the jobs
                             (default is "NORMAL" corresponding to
                             the sequence TERM,HUP,INT,KILL )
                             This can be also set with property NC_STOP_SIGNALS
                             or with the environment variables NC_STOP_SIGNALS
                             or VOV_STOP_SIGNALS.
                             Priority: 1. Option -signal
                                       2. property NC_STOP_SIGNALS
                                       3. env variable VOV_STOP_SIGNALS
                                       4. env variable NC_STOP_SIGNALS
                                       5. default
                             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.
      -after <s>          -- Start sending signals after specified seconds.
                             This is an initial delay, between 0 and 20s.
      -delay  <s>         -- Minimum delay between signals (in seconds),
                             between 0 and 20s. Default is 1.
                             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 VOV_STOP_SIGNAL_DELAY will be used.
                             Priority: 1. Option -delay
                                       2. property NC_STOP_SIG_DELAY
                                       3. env variable VOV_STOP_SIGNAL_DELAY
                                       4. env variable NC_STOP_SIG_DELAY
                                       5. default
      -reason <reason>    -- Give a reason for the stop. This is stored on a property
                             attached to the stopped jobs.

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

```
SEE ALSO:
    % vovshow -env VOV_STOP_SIGNALS
    % vovshow -env NC_STOP_SIGNALS
    % vovshow -env VOV_STOP_SIGNAL_DELAY
    % vovshow -env NC_STOP_SIG_DELAY
```

7.2.1 Override Signals to Stop a Job

A job can be stopped by overriding the sequence of signals that are sent for the job. To do so, set the properties `NC_STOP_SIGNALS` and `NC_STOP_SIG_DELAY`. For more information, refer to the usage information above.

7.2.2 Automatic Stopping Based on Elapsed Time

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

Example:

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

7.2.3 Automatic Stopping Based on CPU Time

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

Example:

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

7.3 Forgetting Jobs

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

- Completed jobs are forgotten after one hour.
- Failed and Idle jobs are forgotten after two days.
- Queued and Running jobs are never forgotten.

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

Usage: `'wx forget'`

```
wx: Usage Message

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

  USAGE:
      % wx forget [OPTIONS] <jobId> ...

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

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

Details about the command options are listed below:

Option	Description
-age *age*	Forget all my jobs older than the specified age, expressed as a time specification.
-dir *dirname*	Forget all the jobs in the current directory.
-subdirs *dirname*	Forget all the jobs in the current directory and in all its subdirectories.
-h	Show the help message.
-mine	Forget all my jobs.
-n	Forget all my jobs older than on day.
-selrule *rule*	Use a selection rule to forget only the jobs that meet a specific criteria.

Note: a USER can only have the own jobs forgotten. ADMIN can have other jobs forgotten.

Examples:

```
% wx forget 2345 2355
wx: message: Forgetting 2 jobs
% wx forget -mine
wx: message: Forgetting 40 jobs
```

7.4 Cleanup Log Files

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

```
% wx clean
```

Usage: `'wx clean -h'`

```
wx: Usage Message

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

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

  USAGE:
       % wx clean [OPTIONS] [LIST_OF_DIRS]

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

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

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

```
% wx clean -R
```

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

```
% wx clean -nozap
```

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

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

To have WorkloadXelerator automatically run `wx clean` every day, you can schedule a periodic job to do that. For example, the following command schedules a cleanup once a day in the current directory:

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

8 Debugging Jobs without Running WorkloadXelerator

8.1 Debug by Running Jobs without WorkloadXelerator

On occasion, jobs that run successfully outside of WorkloadXelerator fail when run through WorkloadXelerator. When this occurs, mostly likely the setups are not the same: the environment, inputs or other parameters may be different, a misconfiguration or there is a problem with NFS. To resolve such issues, using the command `wx debug` can show you the steps that WorkloadXelerator takes to run the job.

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

8.2 How to Use wx debug

Usage: `'wx debug'`

```
wx: Usage Message

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

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

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

      USAGE:
             % wx debug <jobId>

      OPTIONS:
        -h                  - Show this message
```

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

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

Example:

```
% wx debug 01597942

# This job was run on host bear. To run the same
# job without going through WorkloadXelerator, please follow these steps:

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

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

# 3. Switch to the environment
ves SYNOPSYS

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

9 Migrating from LSF

9.1 LSF Emulation

This section is intended for those who have been using the LSF batch system natively and are now moving to using WorkloadXelerator on top of the LSF batch system.

The Runtime software installation includes scripts that are designed to minimize the impact of transitioning workloads from using LSF directly and instead using the WX hierarchical scheduler. Ideally, the use of these scripts, will enable the redirection of workloads from using an LSF cluster directly to one using a WX scheduler in front of the same LSF cluster. While the scripts cover the common usage of LSF, higher performance and a richer set of scheduling directives will be achieved with using the native WX command set.

The emulated commands include the following:

- bhist
- bhosts
- bjobs
- bkill
- bmgroup
- bpeek
- bqueues
- bstat
- bsub
- lshosts
- lsid

The scripts that emulate the above commands are available in the directory `$VOVDIR/scripts/lsfemulation`, which is not in the PATH in the default setup. This setup avoids collision with the platform scripts and commands.

The emulated commands are added to the path by adding the Runtime named environment `LSFEMUL`. The `LSFEMUL` environment setup is installed in `$VOVDIR/etc/environments/LSFEMUL.*`.

```
% ves +LSFEMUL
% bsub sleep 10
% bjobs -a
```

9.1.1 Configuring Resource Mapping

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

To map resources from one system to the other, customize the following file:
`$VOVDIR/local/lsfemulation/config.bsub.tcl`
This file contains a set of assignments to the Tcl arrays `MAP_LSF2NC()` and `MAP_RUSAGE()`.
Note: This file is used only by the Runtime `bsub` emulator.

An example for `config.bsub.tcl` can be found in the usual location for configuration files, a subdirectory of $VOVDIR/etc/config: $VOVDIR/etc/config/lsfemulation/config.bsub.tcl.

File: $VOVDIR/etc/config/lsfemulation/config.bsub.tcl

```
#
# Sample configuration of the bsub emulation.
#
# This file must be placed in $VOVDIR/local/lsfemulation/config.bsub.tcl
#

# If you want support for exclusive access to machines (option -x)
# you need to:
# 1. Uncomment the line below 'set bsubopt(percent) 1'
# 2. Make sure all slaves offer the resource PERCENT/100
# 3. Make sure all jobs request at least PERCENT/1 (see vnc_policy.tcl)
set bsubctrl(percent) 1

# Set this to 1 to cause bsub to always send email
set bsubctrl(alwaysmail) 1

# Truncate emailed log files after this many bytes
# negative values (e.g. -1) mean mail the whole file (BEWARE)
# zero means accept the default (65536 bytes)
set bsubctrl(logmax) 0

# Emulation transforms -m hostnames into an OR expression,
# which can slow down the NC scheduler if too complex.
# Hosts after this count are silently dropped to avoid slow scheduling
set bsubctrl(moptmax) 6

# Map LSF 'select' resources into NC resources.
# select[rhel4]  -> "linux"
set MAP_LSF2NC(rhel3) "linux"
set MAP_LSF2NC(rhel4) "linux"
set MAP_LSF2NC(rhel5) "linux"
set MAP_LSF2NC(RH4_64) "linux x86_64"

# Map LSF 'rusage' resources into NC resources,
# typically resources of type License:
# Example:
#  rusage[dc=1]   -> "License:Design-Compiler"

set MAP_RUSAGE(dc)   "License:Design-Compiler"
set MAP_RUSAGE(pt)   "License:PrimeTime"
set MAP_RUSAGE(drc)  "License:lic_drc"
set MAP_RUSAGE(lvs)  "License:lic_lvs"
set MAP_RUSAGE(erc)  "License:lic_erc"
```

9.1.2 Emulating the LSF Report in the Output Log

Some legacy scripts expect some LSF specific lines in the log file of a job. This can be achieved with a post-command that adds those lines to the log. An example of such command is `post_job_report.csh`

File: /remote/release/VOV/2016.09u20_68817_Nov02/linux64/etc/post/post_job_report.csh

```
#!/bin/csh -f
# -*- Tcl -*- \
   exec vovsh -f $0 $*:q

set usage "
Description:
  post_job_report.csh

  Used for some jobs submitted with the bsub emulator:

Example:
% ves +LSFEMUL
% bsub -Ep $VOVDIR/etc/post/post_job_report.csh -J test.lsf_with_jobreport cal 2015
"

if { $argv == {} } {
    VovPrintUsage $usage
}

source $env(VOVDIR)/tcl/vtcl/vovlsfemulib.tcl

set jobId       [lindex $argv 0]
set logFileName [lsfEmuGetJobLogFileName $jobId]
set report      [lsfEmuFmtJobReport $jobId]

if { $logFileName ne "" } {
    VovMessage "Adding job report to $logFileName"
    set fp [open $logFileName "a"]
    puts $fp [lsfEmuFmtJobReport $jobId]
    close $fp
} else {
    VovMessage "No log file found for job $jobId\n$report"

    set whyOld ""
    set whyNew "Cannot find a log file for this job $jobId"
    if { [catch {set whyOld [vtk_prop_get $jobId WHY]}] } {
        set why $whyNew
    } else {
        set why "$whyOld\n$whyNew\n$report"
    }
    catch {vtk_prop_set $jobId WHY $why}
}

exit 0
```

The post command can be specified with the option -Ep of the `bsub` emulator. Example:

```
% bsub -Ep $VOVDIR/etc/post/post_job_report.csh   [OTHER OPTIONS]
... command
```

9.1.3 Debugging the LSF Emulation Layer Usage

To debug as well as test and verify an LSF emulation script, it can be helpful to view the issued commands and the used options and values.

If the environment is set with variable `VOV_LOG_LSFEMUL` to the name of a file, all emulation commands will be logged in that file. An example follows:

```
% setenv VOV_LOG_LSFEMUL ~/lsfemul.log
% bsub sleep 11
% lsid
% cat $VOV_LOG_LSFEMUL
```

10 Legal

10.1 Runtime Inc Copyright

Information in this document is subject to change without notice and does not represent a commitment on the part of Runtime Inc. The software described in this document is furnished under a license agreement. The software may be used only in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, or translated into any language in any form by any means, without the written permission of Runtime Inc.

Portions of Runtime Inc. technology are covered by U.S. Patents 5,634,056 7,937,706 and 9,658,893. Other patents pending.

Runtime Inc™, FlowMaker™, FlowRunner™, FlowTracer™, HERO™, LicenseAllocator™, LicenseMonitor™, MultiQueue™, NetworkComputer™, ResourceMonitor™, WorkloadXelerator™, and WorkloadAnalyzer™ are trademarks of Runtime Inc.

Other products mentioned are trademarks or registered trademarks of their respective companies.

10.1.1 ArgumentParser

A slimline C++ class for parsing command-line arguments

10.1.2 Boost

Portions of the Boost collection of C++ libraries are used in certain Runtime software to aid in software portability across platforms.

derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

10.1.3 Fossil SCM

The Fossil software configuration management tool is used in certain Runtime software to provide versioning capabilities for various configuration files. Fossil is released under a 2-clause BSD license:

Copyright 2007 D. Richard Hipp. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and contributors and should not be interpreted as representing official policies, either expressed or implied, of anybody else.

10.1.4 GD Graphics Library

The built-in graphics capabilities of the software take advantage of the GD Graphics Library, which is available from http://www.boutell.com/gd. More detail on the GD Graphics Library is available at http://www.boutell.com/gd/index.html on the http://www.boutell.com web site.

The GD Graphics Library distribution used is *gd-2.0.33*.

10.1.5 GNU Utilities for Windows

The LicenseMonitor™ LicenseManager functionality employs CVS (Concurrent Versions System) to maintain a history of changes to licensing files. It also makes use of the file program in order to determine file types. This utility requires the magic, regex, and zlib DLLs on Windows. On Unix-based systems, these utilities and their required libraries are normally already present in the operating system loadset, but they are not normally present on Windows systems. The cvs, file, and gzip programs are included in the Windows distribution of Runtime software, as are the aforementioned DLLs that are required for the file program.

The distribution of CVS included in this version of Runtime software is 1.11.22. The distribution of file (and magic) included in this version of Runtime software is 5.03.3414.

The distribution of regex2 included in this version of Runtime software is 2.7.2853. The distribution of zlib1 included in this version of Runtime software is 1.2.8.

The distribution of gzip included in this version of Runtime software is 1.2.4.

All of these software components are released under the GNU Public License (GPL).

10.1.6 Graphviz -- Graphical Visualization Software

Runtime software makes use of Graphviz libraries as part of its console Graphical User Interface (GUI). The version of Graphviz distributed with this version of Runtime software is 2.38.0. The Graphviz license can be viewed at http://www.graphviz.org/License.php.

Runtime has modified the Graphviz libraries for its use with Runtime products. To obtain a copy of the modified Graphviz libraries, please contact support@rtda.com.

10.1.7 [incr Tcl]

The [incr Tcl] software is used in constructing single-file distributables of Runtime software. The distribution of [incr Tcl] included in this version of Runtime software is that which is included in the TclKit 1.8.5 distribution. [incr Tcl] is licensed under a BSD-style license:

> This software is copyrighted by Lucent Technologies, Inc., and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.
>
> The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.
>
> IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
>
> THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

10.1.8 Metakit

The Metakit software is used in constructing single-file distributables of Runtime software. The distribution of Metakit included in this version of Runtime software is that which is included in the TclKit 1.8.5 distribution. Metakit is licensed under an MIT-style license:

> Copyright (c) 1996-2007 Jean-Claude Wippler
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

10.1.9 mysqltcl

Runtime Inc makes use of the mysqltcl MySQL Tcl Interface library. Distribution version: 3.05.

> Copyright (c) 1994, 1995 Hakan Soderstrom, Enskede, Sweden and Tom Poindexter, Denver, Colorado
>
> Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice and this permission notice appear in all copies of the software and related documentation.
>
> THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
>
> IN NO EVENT SHALL HAKAN SODERSTROM OR SODERSTROM PROGRAMVARUVERKSTAD AB BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

10.1.10 nginx

The nginx software is used by Runtime software to provide the entry point for HTTP/HTTPS communications between the Runtime web server and remote web clients. The distribution of nginx included in this version of Runtime software is 1.9.2. nginx is licensed under a BSD-style license.

10.1.11 Octtools

This software also uses packages from the Octtools-5.1 distribution from UC Berkeley, namely: *errtrap options st timer utility*

Octtools are covered by the following copyright notice:

> Oct Tools Distribution 5.1
>
> Copyright © 1988, 1989, 1990, 1991 Regents of the University of California.
> All rights reserved.
>
> Use and copying of this software and preparation of derivative works based upon this software are permitted. However, any distribution of this software or derivative works must include the above copyright notice.
>
> This software is made available AS IS, and neither the Electronics Research Laboratory or the University of California make any warranty about the software, its performance or its conformity to any specification.
>
> Suggestions, comments, or improvements are welcome and should be addressed to:
>
> octtools@eros.berkeley.edu

These packages have been developed at UC Berkeley from 1985 to 1991 by the Berkeley CAD group. Special thanks to David Harrison, Tom Laidig, Peter Moore, Richard Rudell, Rick Spickelmeir.

10.1.12 OpenLDAP

The OpenLDAP client library is utilized by the vovserver binary to perform LDAP-based authentication. OpenLDAP software is released under the OpenLDAP Public License:

> The OpenLDAP Public License Version 2.8, 17 August 2003
>
> Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:
>
> 1. Redistributions in source form must retain copyright statements and notices,

2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and

3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

10.1.13 OpenSSL

Runtime software makes use of OpenSSL libraries to provide secure communications to various services. The version of OpenSSL distributed with this version of Runtime software on Unix-based platforms is 1.0.2a, and on Windows, is 1.0.2e. The OpenSSL license can be viewed at https://www.openssl.org/source/license.html.

10.1.14 pgtcl-ng

Runtime software makes use of the pgtcl-ng PostgreSQL Tcl Interface library. Distribution version: 2.0.0.

This is the license for pgtcl-ng:

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

10.1.15 PhantomJS

Runtime software makes use of the PhantomJS utility for parsing HTML files and working with JavaScript objects from the command line. Distribution version: 2.1.1.

> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
>
> * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of the <organization> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
>
> THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10.1.16 PostgreSQL

Runtime software makes use of PostgreSQL database for its back-end data storage reporting needs. PostgreSQL database is released under the PostgreSQL License. Distribution version: 9.6.1.

> PostgreSQL is released under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT licenses.
>
> PostgreSQL Database Management System (formerly known as Postgres, then as Postgres95)
>
> Portions Copyright (c) 1996-2010, The PostgreSQL Global Development Group
>
> Portions Copyright (c) 1994, The Regents of the University of California
>
> Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.
>
> IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
>
> THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

10.1.17 Reprise

This software includes and ships with Reprise Software Licensing Manager (RLM) binaries (v9.3rel) which are used to validate Runtime Licenses.

> Copyright © 2006-2011, Reprise Software, Inc. All Rights Reserved.
>
> Reprise License Manager, OpenUsage, and Transparent License Policy are all trademarks of Reprise Software, Inc. RLM contains software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org) Copyright © 1998-2003 The OpenSSL Project. All rights reserved.

10.1.18 SQLite

The SQLite database is public domain as described in http://www.sqlite.org/copyright.html. Distribution version: 3.8.3.

10.1.19 Tcl/Tk

The graphical user interface is implemented with Tcl 8.6.5/Tk 8.6.5.

Tcl/Tk includes the following copyright notice:

> This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.
>
> The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.
>
> IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
>
> THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
>
> GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 © (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 © (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

10.1.20 TclVfs

The TclVfs software is used in constructing single-file distributables of Runtime software. The distribution of TclVfs included in this version of Runtime software is that which is included in the TclKit 1.8.5 distribution. TclVfs is licensed under a BSD-style license:

> This software is copyrighted by the Vince Darley, and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.
>
> The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.
>
> IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
>
> THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE

AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

10.1.21 tcpkill

The utility 'tcpkill' has been included in the release, under the name 'vovtcpkill'. The utility comes with this copyright notice.

Copyright (c) 1999-2010 Dug Song <dugsong@monkey.org>, et al. All rights reserved, all wrongs reversed.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The names of authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10.1.22 TkConsole

Copyright 1995-2002 Jeffrey Hobbs, jeff(a)hobbs(.)org

Release Info: v2.4, CVS v1.82 2004/11/11 17:22:13

Documentation available at: http://tkcon.sourceforge.net

10.1.23 TWAPI

On Windows, the TWAPI Tcl library is used to interface with the Windows API for various functions, such as service management. Distribution version: 3.1.17.

TWAPI includes the following copyright notice:

Copyright (c) 2003-2008, Ashok P. Nadkarni
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- The name of the copyright holder and any other contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10.1.24 XYNTService

The XYNTService utility is a general-purpose Windows service wrapper that allows for the creation of custom services for programs that do not have native service support. This utility is only included in the Windows distribution of Runtime software. The distribution of XYNTService included in this version of Runtime software is dated 02.22.2008 and is released under the Code Project Open License (CPOL).

10.1.25 Zlib

The 'zlib' compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data. Distribution version: 1.2.8.

Zlib includes the following copyright notice:

> Copyright © 1995-1998 Jean-loup Gailly and Mark Adler
>
> This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.
>
> Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions: 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 3. This notice may not be removed or altered from any source distribution.
>
> Jean-loup Gailly jloup@gzip.org
> Mark Adler madler@alumni.caltech.edu
>
> The data format used by the zlib library is described by RFCs (Request for Comments) 1950 to 1952 in the files http://www.ietf.org/rfc/rfc1950.txt (zlib format), http://www.ietf.org/rfc/rfc1951.txt (deflate format) and http://www.ietf.org/rfc/rfc1952.txt (gzip format).