# WorkloadXelerator™ Administrator 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 Scope of this Administrator Guide for WX

This guide is written for the WorkloadXelerator (WX) system administrator who needs to configure and manage one or more WX instances.

The administrator is expected to understand Unix system processes, the dynamics of Unix interactive shells, shell scripting techniques, and general trouble-shooting concepts. Some knowledge of schedulers is also expected.

For details about the usage and capabilities of using WorkloadXelerator, referring to the WorkloadXelerator User Guide is recommended.

### 1.1.1 Related Documents

Since WX is a derivative of NC (NetworkComputer), most of the concepts explained in the NetworkComputer Admin Guide also apply to WX, even though they are not explained in this guide.

The following documents provide additional information that is related to using and configuring WorkloadXelerator:

- RTDA Software Installation to learn how to install WX and the rest of the RTDA software
- WorkloadXelerator User Guide to learn the end-user point of view for WX
- NetworkComputer Admin Guide to learn all the advanced tricks that also apply to NC and WX

- VOV Subsystem Reference Manual contains the reference material about projects, files, registry, etc.

## 1.2 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.

### 1.2.1 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.

### 1.2.2 Examples of Modes of Operation

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

#### 1.2.2.1 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.

#### 1.2.2.2 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.

#### 1.2.2.3 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.

### 1.2.3 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.

## 1.3 WorkloadXelerator vs. NetworkComputer

The two products are essentially the same, but serve rather different usage models. The following table clarifies the differences between them.

| Item | NetworkComputer | WorkloadXelerator | Comments |
|---|---|---|---|
| Manager command to start and stop | ncmgr | wxmgr | Essentially the same command. |
| User command submit workload | nc | wx | Essentially the same command. |
| Slaves | Static set of slaves, fully | Elastic set of slaves, determined by | |

| | | | |
|---|---|---|---|
| | managed: Each slave monitors load and resources on its host | workload; each slave has one slot and is available regardless of load on host | |
| Licensing of VovServer | server_nc | server_wx | |
| Licensing of VovSlave | slots_nc | slots_wx | |
| Job Resources | -no change- | The resource expression is silently augmented to be "`Bucket:<BucketId> OR ( <resource_expression>)`". | In WX, we want to allow precise routing of jobs to the slaves that have been created for that specific bucket. This is for information only. The user does not see this change. |
| Job Placement Policy (JPP) | All JPP are supported: fastest, smallest, largest, slowest, first, ... | Only JPP==first is supported. The JPP of the workload is passed to NC if base scheduler is NC | |
| Preemption | Full preemption support. | Fully supported but normally not used: job modulation is preferred | |
| Blackhole Detection | Supported: slaves are allowed to recover | Supported: slaves are dismissed if they appear to be black-holes. | |
| Reservations (subject to change in future releases) | Only ADMIN can create reservations | USER can create reservations | In WX we allow users to reserve slaves for specific buckets |

## 1.4 Starting up WorkloadXelerator

The command to start and stop WorkloadXelerator is `wxmgr`, in all ways similar to `ncmgr`.

The default WorkloadXelerator queue is named *wx*.

```
% wxmgr start
wxmgr: message: Checking the license...
wxmgr: message: ... the license is good.
wxmgr: message: Starting WorkloadXelerator
wxmgr: message:   with name    wx
wxmgr: message:   on host      hostname
wxmgr: message:   in directory /remote/release/vov/vnc
wxmgr: message:   as user      username
Do you want to proceed? (yes/[no]) > yes
wxmgr: message: Updating config file '/remote/release/vov/...../vwx.tcl'
wxmgr: message: Waiting for server to be ready ...
wxmgr: message: Sanity check...
wxmgr: message: WorkloadXelerator wx@hostname is ready.
wxmgr: message: Sanity check...
wxmgr: message: Confirming start of required daemons...
wxmgr: message:
wxmgr: message: wx@mac01ac is ready. To access the web UI:
wxmgr: message:
wxmgr: message:   http://mac01ac:6439    -- Requires login
wxmgr: message:
wxmgr: message:  On C-Shell:  setenv NC_QUEUE /remote/release/vov/wx/wx.swd/setup.tcl
wxmgr: message:  On Bash:     export NC_QUEUE=/remote/release/vov/wx/wx.swd/setup.tcl
```

The command below is equivalent to `wxmgr start` above, but with the explicit values of the working directory, port number and of and the queue name. You will use a command similar to this to start other WX instances.

```
% wxmgr start -queue wx -port 6578 -dir $VOVDIR/../../wx
...output omitted...
```

Once the queue has started, you need to be able to connect to it. This is done by setting the environment NC_QUEUE (Attention: same variable used in NetworkComputer! Not WX_QUEUE). This variable should be set to the setup.tcl file that has been created in the server working directory of the new WX instance. These values are shown clearly at the end of the start log (see above).

```
## Example for C-shell users
% setenv NC_QUEUE $VOVDIR/../../wx/wx.swd/setup.tcl
```

Now we can check that the WX is responding:

```
% wx hosts
  # SLAVE       LOAD STATUS   JOBS      HB RESERVE             MESSAGE
  1 WXLauncher  0.00 ready    0/8      43s C=LauncherClass 1y334d Licensed for 8 slots
```

This shows that there is already one vovslave called "WXLauncher" which is reserved for a special class of jobs called "LauncherClass". This slave is used to issue the requests to the underlying base scheduler on behalf of the users who submit workload.

Now the WX instance needs to be connected to the base scheduler. This can be done most easily by using the utility `vovwxconnect`. In the following example, we connect this new WX instance to the main NetworkComputer instance called "vnc"

```
% wx cmd vovwxconnect  -nc vnc
```

vovwxconnect creates the appropriate daemon configuration file (vovwxd/config.tcl) and starts the daemon.

**Usage: `'vovwxconnect'`**

```
vovwxconnect: Usage Message

  DESCRIPTION:
      Utility to setup a connector for WX to a primary queue

  OPTIONS:
      -h
      -v
      -nc NCQUEUENAME  -- Connect with specified NC queue.
      -vovdir PATH     -- Specify VOVDIR for NC queue, if different than that of WX.
      -lsf             -- Connect with LSF.
      -lsfemul NCQ     -- Connect with NC but using the LSF emulation interface (only for debugging)
      -pbs             -- Connect with PBS
      -pbsemul NCQ     -- Connect with NC but using the PBS emulation interface (only for debugging)
      -legacy          -- Connect using legacy daemons vovelasticd vovlsfd.
      -show            -- Show what is connected.
      -test            -- Validate connections by running a test job.
      -nostart         -- Do not start the vovwxd daemon.
      -loglevel        -- Set verbocity level for vovwxd (0-6)

  EXAMPLES:
     % vovproject enable MY_WX_PROJECT
     % vovwxconnect -nc vnc_test
     % vovwxconnect -nc vnc_test -vovdir /some/path/to/a/vovdir/used/by/vnc_test
     % vovwxconnect -pbs
     % vovwxconnect -lsf
     % vovwxconnect -test
     % vovwxconnect -lsfemul vnc_test
```

### 1.4.1 Security

The security file `wx.swd/security.tcl` determines who may connect to the system and from which host. By default, the owner of the VOV server has ADMIN privilege from all hosts. Check the file `wx.swd/security.tcl` to verify that the login name is correct and add any others needed.

Check the file `wx.swd/security.tcl` to verify that the following lines exist:

```
# All users can connect from anywhere
vtk_security + USER +
# You want to be ADMIN
vtk_security YOURNAME ADMIN +
# Root needs to have ADMIN privileges, to permit vovslaveroot to
connect
vtk_security root ADMIN +
# The cadmgrs VovUserGroup
 needs to have ADMIN privileges
vtk_security -group cadmgrs ADMIN +
```

Check the WorkloadXelerator system status with the command:

```
% wx cmd vsi

Vov Server Information - 03/23/2018 17:04:04

wx@mac01ac:6439                    | URL: http://mac01ac:6439
-------------------------------------------------------------------------
 Jobs:                           1 | Workload:
 Files:                          2 |   - running:                     0
 Sets:                          18 |   - queued:                      0
 Retraces:                       0 |   - done:                        0
                                   |   - failed:                      0
-------------------------------------------------------------------------
 Slaves:                         1 | Buckets:                         0
  - ready:                       1 | Duration:                       0s
 Slots:                          8 | SchedulerTime:                0.00s
-------------------------------------------------------------------------
 TotalResources:             1,685 | Pid:                         51213
                                   | Saved:                  26m41s ago
                                   | Size:                     16.00MB
                                   | TimeTolerance:                  3s
-------------------------------------------------------------------------
```

Submit the first job to WX:

```
# Open the GUI to monitor the job.
% wx gui &
% wx run sleep 30
```

After a few seconds, you will see the job turn from cyan to orange and then eventually green to represent successful execution.

If either the server are not running, refer to the troubleshooting guide.

## 1.5 Migrating from the previous version

When WX was introduced in version 2016.09, it was based on Tcl daemons called "vovelasticd" and "vovlsfd". This section describes how to migrate from the previous version to the new version based on the binary daemon "vovwxd".

Let's assume that the WX instance is called "wx" and the NC instance is called "vnc", the default name. Let's start with the normal setup:

```
% vovproject enable wx
% vovwxconnect -nc vnc
% vovdaemonmgr show vovwxd
DOWN
% cd `vovserverdir -p vovwxd`
```

In the directory WX.swd/vovwxd you can find the log file for vovwxd. You can easily look at that file to see if the daemon is up and running.

Upon starting, vovwxd checks the WXSWD/autostart directory for scripts to start legacy daemons, i.e. vovelasticd or vovlsfd. If such legacy autostart scripts exist, vovwxd prints a warning stating that the vovwxd autostart script will not be installed. The legacy daemon will need to be manually stopped, and its autostart script will need to be disabled (renamed or removed). The vovwxd autostart script will then need to be copied from $VOVDIR/etc/autostart into WXSWD/autostart.

```
% cp $VOVDIR/etc/autostart/start_wx_daemon.tcl `vovserverdir -p autostart/.`
```

If a legacy daemon autostart script does not exist, the autostart script for the new vovwxd daemon will be installed into WXSWD/autostart and the daemon will be automatically started.

Copy the configuration file template from $VOVDIR/etc/config/vovwxd/config.tcl into WXSWD/vovwxd.

```
% cp $VOVDIR/etc/config/vovwxd/config.tcl `vovserverdir -p vovwxd/.`
```

(Temporary) Copy the batch system driver script from $VOVDIR/etc/config/vovwxd/vovnc.tcl into WXSWD/vovwxd.

```
% cp $VOVDIR/etc/config/vovwxd/vovnc.tcl `vovserverdir -p vovwxd/.`
```

Modify `WXSWD/vovwxd/config.tcl` to specify the base queue name(s) and ensure the correct driver script is specified:

```
# Fragment of vovwxd/vconfig.tcl
set CONFIG(driver_script) "vovnc.tcl"
set CONFIG(queues) "base_NC_queue_name"
```

If multiple NC queues are to be used as the back-end, specify them in space-separated format:

```
# Fragment of vovwxd/config.tcl
set CONFIG(driver_script) "vovnc.tcl"
set CONFIG(queues) "vnc1 vnc2 vnc3"
```

The vovwxd daemon automatically reads the configuration after changes.

When upgrading WX from version 2016.09, make sure a LauncherSlave is defined. Edit the WX WXSWD/slaves.tcl to add a local slave that will be responsible for submitting slaves. This slave should be reserved for the LauncherClass class and provide a resource called WXLauncher. For example:

```
# Fragment of WX.swd/slaves.tcl
vtk_slave_define localhost -name "WXLauncher" -resources "WXLauncher"

#
# Create a reservation for the slave.
# The syntax is as follows:
# vtk_reservation_create type what quantity start end <OPTIONS>
#
 vtk_reservation_create slave WXLauncher 1 0 forever -jobclass LauncherClass
```

# 2 WorkloadXelerator Server and System Configuration

## 2.1 Starting and Stopping WorkloadXelerator

The command `wxmgr` is identical to `ncmgr`, so to understand the details you can refer to vncmgr.html; just remember to replace "wxmgr" wherever you see "ncmgr".

### 2.1.1 Starting WorkloadXelerator at System Boot Time on Linux

This step is optional.

The instructions in this section are valid for Linux. **Note:** This part of the installation requires root permission.

The WorkloadXelerator VovServer can be restarted at reboot by installing the proper script in both the `/etc/rc3.d` and `/etc/rc5.d` directories for Linux.

Run the following commands on the host that was selected as the WorkloadXelerator VovServer.

```
% /bin/su -
% cp $VOVDIR/etc/boot/S99wx /etc/rc3.d/S99wx
% chmod 755 /etc/rc3.d/S99wx
% vi /etc/rc3.d/S99wx
....
Edit configurable items as needed.
```

**Note:** `sudo` should be used where configured. To avoid *Trojan Horse* programs, `su` should always be called using the full path `/bin/su` on Linux.

You can test with:

```
% ./S99wx start
```

```
% ./S99wx stop
```

**Note:** Re-start the WorkloadXelerator server with the command `./S99wx start` after testing the `stop` capability.

## 2.2 Daemons for WorkloadXelerator

Some key functionality in WorkloadXelerator is provided by external daemons, as described in the table below.

| Daemon | Who needs it? | Description |
|---|---|---|
| `vovwxd` | The binary for running WorkloadXelerator on top of NetworkComputer. | This daemon is required to interface the WorkloadXelerator system to an existing NetworkComputer system. More info... |
| `vovelasticd (obsolete)` | Only if you are running WorkloadXelerator on top of NetworkComputer | This daemon is required to interface the WorkloadXelerator system to an existing NetworkComputer system. More info... |

The status of the daemons can be viewed at the page `/cgi/daemons.cgi`, or can be shown with this command:

```
% wx cmd vovdaemonmgr show
vovresourced             OK
vovdbd                   OK
vovwxd                   OK
```

### 2.2.1 WorkloadXelerator-on-NetworkComputer Theory of Operation & Trouble-shooting

This section describes the theory of the operation of WorkloadXelerator-on-NetworkComputer and how to address problems should they occur.

### 2.2.2 Theory of Operation: Connecting WorkloadXelerator to NetworkComputer

In general, WorkloadXelerator connects to a base scheduler with an elastic grid daemon. About connecting to the NetworkComputer:

WorkloadXelerator connects to NetworkComputer with the daemon `vovwxd`. The run directory for this daemon is `.swd/vovwxd`. The three significant files are `vovwxd.log`, `config.tcl`, and `vovnc.tcl`.

- The `config.tcl` file allows the elastic parameters to be configured.
- A fresh timestamp on the log file confirms that the elastic daemon is running.
  **Note:** It is essential that the elastic daemon is running for jobs to be submitted to the base scheduler.

A change to the `config.tcl` file is automatically picked up by the vovwxd. A crashed or halted daemon can be restarted with the command

```
% wx cmd vovdaemonmgr start
vovresourced             ALREADY RUNNING
vovdbd                   ALREADY RUNNING
vovwxd                   STARTING...
```

The vovwxd daemon watches the WorkloadXelerator workload; if some bucket is waiting for hardware or software resources, then the daemon issues a request to the base scheduler for more resources.

When the underlying scheduler dispatches this job to an execution host, the job fires a new vovslave that connects back to the WX, therefore advertising new computing resources that can be used to process the jobs queued in WX.

When this occurs, you will see an extra box appear in the LED monitor of the vovconsole (located on the upper bar of the vovconsole main window). The slaves started by an elastic daemon appear as additional boxes. The box size decreases as more slaves are added.

**Key notes:**

- `vovwxd` sends "slave job requests" to the base scheduler; the actual workload in WX stays in WX and is executed as the base scheduler satisfies the requests from `vovwxd`
- The base scheduler only sees slave requests. It has no direct visibility into the details of the WX work load, although the slave's resource request (license, limits, memory, cores etc) precisely matches the underlying workload.
- To WorkloadXelerator, it sees a "private" pool of slaves to which it can submit jobs. Being elastic, the pool may grow or shrink.

- Typically, the slaves requested by vovwxd offer only one slot, so that only one job at a time can be run on one of these slaves.

WX slaves terminate when one of two conditions is met:

1. the maximum idle time is exceeded. This is controlled by the parameter "Max Idle Time" represented by the variable CONFIG(slave,maxidle) in config.tcl, which is typically 30s;

2. the maximum life is exceeded and the slave is idle. This is controlled by the parameter CONFIG(slave,maxlife) and has a default value of 1w (1 week), but typical values can be 1h or 2h. This allows fair share policies on the underlying queue to be respected by keeping the maximum job duration in NC to some reasonable range.

When a **maximum idle is exceeded**, the slave stops accepting jobs by declaring itself to be "DONE"; the slave turns blue in the slave monitor status column. After a few seconds the slave exits and the entry for it disappears from the monitor. At that time, the job representing the request for that slave will terminate in NetworkComputer. When the **maximum life is exceeded**, the slave declares itself as suspended (purple) and no new jobs are sent to it, while all jobs currently running on that suspended slaves are allowed to continue running and to terminate normally. The slave will exit when no more jobs are running on it. This behavior is essential for maintaining adherence to fair share policies.

A normal WX slave will go through the following states:

| | |
|---|---|
| REQUESTED | Request for a new slave resource has been issued to base scheduler. |
| NOLIC | Slave is connecting and is waiting to be given a license. |
| READY | Slave is ready to receive a job. |
| FULL | Slave is running a job. |
| OVERLOADED | The load on the slave is higher than a specified threshold. |
| SUSPENDED | Slave is no longer accepting jobs. |
| PAUSED | Slave has been paused by the base scheduler. |
| DONE | Done, about to be deleted. |

### 2.2.3 When Things Go Well

In WorkloadXelerator's slave monitor, you will see a reasonable number of slaves listed; most are classed as FULL and some as REQUESTED. Normally, each slave will have 1 slot and that will be occupied. For freshly created slaves they may go into a NOLIC state for a brief period while they establish their connection, license and capabilities. Some slaves may be SUSPENDED/purple - they're finishing off the last job they've been allotted before max life kicks in. Some status will be READY/green for a brief period; most will be FULL/yellow (running a job). A few will be red indicating an overload.

From the NetworkComputer perspective, you can look at a set called "WXSlaves:wx" to see all requested jobs coming from vovwxd. Each set should have some number of valid jobs (slaves that have exited normally) and few running/orange jobs (the current work being done) and a small number of queued/cyan jobs. Only a small number of queued jobs should ever be present (e.g. 10-50) even though the WorkloadXelerator session may have several hundred thousand jobs ready to run. The vovwxd daemon will continue to replenish the pending number as they get executed, so that the NetworkComputer queue size remains small. This does not impact fair share negatively as only a single waiting job causes fair share to kick in; teams that use WorkloadXelerator will not be at a disadvantage just because a handful of jobs are present in NetworkComputer at any one time compared with others that run their entire workload in NetworkComputer.

## 2.3 Configuration of vovwxd

In order to successfully run jobs under WX, submitting to NC, some configuration must be done up front.

NC job classes that will be used by by WX must be present in both NC and WX. It is important to note that job classes are processed by WX only, and the slave job submitted to the base queue will contain the post-processed resources, limits, JPP, etc. as defined by the job class. The slave job in NC will be annotated with the job class name for traceability reasons.

Resources that are referenced by jobs submitted to WX must be accessible from the downstream NC queues.

```
==================== Start sample WXSWD/vovwxd/config.tcl file ====================

### vovwxd configuration file

# Specify the driver script to use for interfacing to the back-end batch
# system. For NC, use "vovnc.tcl".
set CONFIG(driver_script)            "vovnc.tcl"

# Specify the space-separated list of back-end queues to which WX should
# submit slave requests.
set CONFIG(queues)                   "vnc"

# Specify the VOV named environment to use for the vovwxd daemon.
# For NC, this will normally be the "BASE" environment.
set CONFIG(cmd,env)                  "BASE"

# Specify a baseline submission command. Normally not needed.
set CONFIG(cmd,submit)               ""

# How often should the vovwxd daemon cycle execute ?
```

```
# The value is a VOV time spec and the default is two seconds.
set CONFIG(refresh)                 5s

# how frequently should we ask for job status from back-end queue ?
# The value is a VOV time spec and the default value is one minute.
set CONFIG(jobstat,checkfreq)       5m

# Remove sick slaves that are older than?
# Value is a VOV time spec, and the default value is five minutes.
set CONFIG(sick,older)              5m

# What is the maximum number slaves we should start?
# Should be set to a high value to enable lots of parallelism.
set CONFIG(slave,max)               500

# What is the maximum number of queued slaves per bucket that we should allow?
set CONFIG(slave,maxQueuedPerBucket) 100

# What is the maximum number of slaves that will be grouped into an array
# for each resource bucket, during each refresh cycle? The absolute max number
# of slaves supersedes this value.
set CONFIG(slave,arrayMax)          0

# What is the longest a vovslave should run before self-exiting?
# Ex: if you set it to 8 hours, and queue 4 3-hour jobs:
# the first slave will run for nine hours (3 x 3-hr > 8-hr) and then exit
# the fourth job will only start when a second slave has been requeued
# and started by the batch execution system.
# This controls the amount of reuse of a slave while it processes jobs.
# To avoid the penalties of:
# noticing a slave is needed
# + submitting to the batch system
# + the batch system to allocate a machine
# You should set this to a high value like a week.
# The value is a VOV time spec
# This is a default value. It can be overridden on a per job basis by putting
# a resource on the job that looks similar to the following.
# MAXlife:1w
set CONFIG(slave,maxlife)       4h

# How long should a slave wait idle for a job to arrive?
# The shorter time, the faster the slot is released to the batch system.
# The longer time, the more chances the slave will be reused.
# The default value is two minutes (usually takes a minute to allocate a
# slot through a batch system).  Value is a VOV time spec
# This is a default value. It can be overridden on a per job basis by putting
# a resource on the job that looks similar to the following.
# MAXidle:2m
set CONFIG(slave,maxidle)       2s

# Are there any extra resources you wish to pass along to the slaves?
# These resources will be passed directly along to the vovslave. They
# are not processed in any way by vovwxd. For example setting
# this to "MAXlife:1w" will not work as you might expect.
set CONFIG(slave,res)           ""

# What is the vovslave update interval for resource calculation?
# Value is a VOV time spec, and the default value is 15 seconds.
set CONFIG(slave,update)        15s

# Do we want to enable debug messages in the vovslave log files?
# 0=no; 1=yes; default=0
set CONFIG(slave,debug)         0

# What level of verbosity should the WX use when writing to its the log file?
# Valid values are 0-7; default=3
set CONFIG(log,level)           6

# What level of verbosity should the vovslave use when writing to its the log file?
# Valid values are 0-4; default=1
set CONFIG(slave,verbose)       1

# How long should the vovslave try to establish the initial connection to the vovserver?
# Values are in seconds, default is 120 seconds.
set CONFIG(slave,timeout)       120


# How much buffer should we consider when adjusting slave,max based on available client connections?
# This number will be subtracted from the available maxNormalClient connections
# Twice this number will be subtracted from the available file descriptors
# Set this number based on how many non-vovslave client connections are anticipated for this session.
set CONFIG(client,derate)       50

# What is the name of the launchers directory? (Default: \"./launchers\")
set CONFIG(launchers,dirname)   "./launchers"

# Remove launchers that are older than?
```

```
# Value is a VOV time spec, and the default value is one hour.
set CONFIG(launchers,older)     1h

# How many slave submissions should be done for each
# job bucket, during each refresh cycle ? (Maximum number of submission commands to be executed)
# This parameter is a replacement for old (slave,maxSubsPerBucket)
set CONFIG(launchers,maxLaunchersPerBucketPerCycle)   99

# Percentage of submissions for newly needed slaves
set CONFIG(launchers,quota)          10

==================== End sample WXSWD/vovwxd/config.tcl file ====================
```

## 2.4 Controlling Access to WorkloadXelerator;

Like all VovServers, access to WorkloadXelerator can be controlled by editing the security file `wx.swd/security.tcl`

For a complete description, look at Security section for NetworkComputer.

### 2.4.1 Example: Restricting WX to a Single User

```
# Example of wx.swd/security.tcl file for a Private WX
vtk_security john     ADMIN +
```

After configuring the security file, WorkloadXelerator must be reset to apply those the changes:

```
% wxmgr reset
```

### 2.4.2 Example: Opening up WX to Many Users

To make WX accessible by all users, assign everyone USER privileges from all hosts, as in this example:

```
# This is an example of the wx.swd/security.tcl file.
# The first + means "everybody"
# The second + means "from all hosts"
vtk_security john ADMIN +
vtk_security +    USER +
```

```
% wxmgr reset
```

## 2.5 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.<br>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.<br>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

## 2.6 Customizing Submission Policies

The job submission behavior of NetworkComputer or WorkloadXelerator can be controlled by the file `vnc_policy.tcl`, which resides in the VovServer configuration directory. This file is used to define the procedures that are listed below.
**Note:** `vnc_policy.tcl` can now reside in **`vnc.swd`**`/vnc_policy.tcl` as well as `$VOVDIR/local/vnc_policy.tcl`.

<table>
<tr><th>Procedure</th><th>Args</th><th>Description</th></tr>
<tr><td>VncPolicyDefaultResources</td><td>{}</td><td>The default resources required by a job.</td></tr>
<tr><td>VncPolicyValidateResources</td><td>{ reslist }</td><td>Ensure that the resource list for a job obeys any number of rules.</td></tr>
<tr><td>VncPolicyValidateEnvironment</td><td>{ envName }</td><td>Make sure that the environment name for a job obeys any number of rules.</td></tr>
<tr><td>VncPolicyValidateCommand</td><td>{ commandLine }</td><td>Make sure that the command line for a job obeys any number of rules.</td></tr>
<tr><td>VncPolicyDefaultPriority</td><td>{ user }</td><td>Assign the default priority to a job based on the user.</td></tr>
<tr><td>VncPolicyUserPriority</td><td>{ user schedPriority }</td><td>Limit the scheduling priority based on the maximum allowed to the user.</td></tr>
<tr><td>VncPolicyUserPriorityExec</td><td>{ user execPriority }</td><td>Limit the execution priority for a job. By default, this returns the priority that has been passed in.</td></tr>
<tr><td>VncPolicyGetJobInfo</td><td>{ key }</td><td>Retrieve job information. Following are the available key values:
<table>
<tr><th>Key</th><th>Description</th></tr>
<tr><td>tool</td><td>Tool or command name, such as hsim</td></tr>
<tr><td>command</td><td>Complete command line (without wrapper)</td></tr>
<tr><td>user</td><td>Login name submitting the job</td></tr>
<tr><td>setName</td><td>Name of the set in which the job is to be placed</td></tr>
<tr><td>group</td><td>The group the submitter requested</td></tr>
<tr><td>inputs</td><td>Inputs to the job</td></tr>
<tr><td>outputs</td><td>Output files of the job</td></tr>
<tr><td>mailuser</td><td>Email address, if notification was requested</td></tr>
<tr><td>wrapper</td><td>Name of the FT wrapper program, such as vw</td></tr>
<tr><td>priority,default</td><td>Default submission priority</td></tr>
<tr><td>priority</td><td>Requested submission priority</td></tr>
<tr><td>resources</td><td>Requested submission resources</td></tr>
<tr><td>env</td><td>Name of requested job run environment</td></tr>
<tr><td>xdur</td><td>Expected duration of job</td></tr>
</table></td></tr>
</table>

These procedures are called at job submission time, and may cause the job entered into the server to have modified resources or priority compared to what the submission requested.

Following is an example for `vnc_policy.tcl`:

**Example vnc_policy.tcl**

```
# This is an example of vnc_policy.tcl
proc VncPolicyDefaultResources {} {
    global env
    return "$env(VOVARCH) RAM/50"
}

proc VncPolicyValidateResources { resList } {
    #
    # This policy adds a minimum RAM requirement
    # for all submitted jobs.
    #
    global VOV_JOB_DESC
    if { $VOV_JOB_DESC(tool) == "vovresgrab" } {
       # Do not touch this type of jobs (see vovresreq).
       return $resList
    }

    if [regexp "RAM/" $resList] {
        # Job already has a RAM constraint.
    } else {
        # Add a RAM constraint.
        lappend resList "RAM/100"
    }
    return $resList
}
```

Following is an example using the tool name. This can be used to send jobs of a certain tool to specific hosts. A Tcl `catch{ }` is used in case someone uses this file with an older version by mistake.

**Fragment of $VOVDIR/local/vnc_policy.tcl**

```
# This is a second example of vnc_policy.tcl
proc VncPolicyDefaultResources {} {
    global env
    return "$env(VOVARCH)"
}

proc VncPolicyValidateResources { resList } {
    #
    # This policy sends tharas jobs to vovslave hosts offering 'tharas_host'
    # and keeps other kinds of jobs off those hosts
    #
    catch {
        set jtool [VncPolicyGetJobInfo tool]
        if { "$jtool" == "tharas" } {
            lappend_no_dup resList tharas_host
        } else {
            lappend_no_dup resList "!tharas_host"
        }
    }
    return $resList
}
```

## 2.7 WorkloadXelerator™ Customization

Many aspects of WorkloadXelerator behavior can be customized:

- Job submission can be controlled with a layer of policies defined in the vnc_policy.tcl file, located either in `wx.swd/vnc_policy.tcl` or in `$VOVDIR/local/vnc_policy.tcl`.
- The WX server behavior can be controlled using the parameters in `wx.swd/policy.tcl`.
- The defaults for `wx run` can be controlled in `$VOVDIR/local/vncrun.config.tcl`. A common use of this file is to disable the automatic check for the validity of the submission working directory.
- The defaults for `wx list` can be controlled in `$VOVDIR/local/vnclist.config.tcl`. A common use of this file is to force list result caching, or to disable listing by job name.
- Some aspects of the GUI can be controlled using `wx.swd/gui.tcl`.

## 2.8 Customizing the wx run Command

### 2.8.1 Configuring Defaults for the 'wx run' Command

The `wx run` command has built-in default features that include checking the validity of the run directory, enabling job profiling, etc.

This section describes how the WX administrator can use the file `$VOVDIR/local/vncrun.config.tcl` to modify some defaults. This file does not exist by default; it must be created when needed.

The defaults are controlled by slots in the `VOV_JOB_DESC` array variable. The `vncrun.config.tcl` file is loaded after the defaults are set; these defaults can be overridden.

For additional information, refer to the section Defining Job Classes for details about VOV_JOB_DESC.

### 2.8.2 Examples

When submitting a job, the default is to check for a logical name (equivalence) for the filesystem where the run directory is located. This is controlled by the 'check,directory' slot.

To change the default to not check the directory, add the following to the `vncrun.config.tcl` file:

```
set VOV_JOB_DESC(check,directory) 0
```

When submitting a job, the default is not collecting profile information, because the data can be large, unless the `-profile` option is used. To make collection of the profile collection the default, add the following to the config file.

```
set VOV_JOB_DESC(profile) 1
```

An example file is included below that shows some other commonly-used settings:

```
# Example content of vncrun.config.tcl
set VOV_JOB_DESC(check,directory) 0

# Other settings that may be useful.
# set VOV_JOB_DESC(priority,default) [VncPolicyUserPriority $username]
# set VOV_JOB_DESC(priority,sched)   $VOV_JOB_DESC(priority,default)
# set VOV_JOB_DESC(priority,exec)    $VOV_JOB_DESC(priority,default)

# set VOV_JOB_DESC(autokill)      0
# set VOV_JOB_DESC(autoforget)    1
# set VOV_JOB_DESC(legalExit)     "0"
# set VOV_JOB_DESC(mailuser)      ""
# set VOV_JOB_DESC(wrapper)       "vw";
# set VOV_JOB_DESC(preemptable)   1
# set VOV_JOB_DESC(profile)       0
# set VOV_JOB_DESC(schedule,date) 0
# set VOV_JOB_DESC(xdur)          -1
```

## 2.9 Customizing the wx list Command

### 2.9.1 Configuring Defaults for the 'wx list' Command

This section describes how the WX administrator can use the file `$VOVDIR/local/vnclist.config.tcl` to modify some defaults for the wx list command. This file does not exist by default; it must be created when needed.

### 2.9.2 Enable list cache

By default, list results are obtained from the server in real-time. In large-scale workload environments, repeated queries can impact server performance. To reduce this impact, a list result cache can be enabled:

```
set NCLIST(cache,enable) 1
```

### 2.9.3 Configure list cache expiration

If the list cache is enabled, list results will be written to a client-side file, and subsequent list requests will be obtained from this file, up to the cache expiration. The default expiration is 30s from creation. After this time, the cache file will be regenerated upon the next list request. To set the cache expiration to a different value:

```
set NCLIST(cache,timeout,default) 60
```

### 2.9.4 Disable Listing by Job Name

Another list operation that can affect server performance in a large-scale workload environment is listing by job name. This is due to the need to compare string values across many jobs. Listing by job name can be disabled entirely, this way:

```
# Fragment of $VOVDIR/local/vnclist.config.tcl
set NCLIST(listbyjobname,enable) 0
```

## 2.10 Troubleshooting

When a problem occurs, first run `vovcheck` and correct the reported errors:

```
% vovcheck
vovcheck: message: Creating report /usr/tmp/vovcheck.report.15765
Test: BasicVariables                                        [OK]
Test: EnvBase                                               [OK]
Test: FlowTracerLicense                                     [OK]
Test: FlowTracerPermissions                                 [OK]
Test: GuiCustomization                                      [OK]
Test: HostPortConflicts                                     [ERROR]
Test: Installation                                          [OK]
Test: OldMakeDefault                                        [OK]
Test: Rhosts                                                [OK]
Test: Rsh                                                   [OK]
Test: SecurityPermissions                                   [WARN]
Test: SlaveRoot                                             [OK]
Test: WritableLocal                                         [WARN]
Test: WritableRegistry                                      [OK]
Test: vovrc                                                 [OK]
vovcheck: message: Detailed report available in /usr/tmp/vovcheck.report.15765
```

### 2.10.1 The Server Does Not Start

The recommended checkpoints:

- Make sure you have a valid RLM license. Type:
  ```
  % rlmstat -a
  ```
- Check if the server for your project is already running on the same machine. Do not start a WX project server more than once. For example, you can try:
  ```
  % vovproject enable project
  % vsi
  ```
- Check if the server is trying to use a port number that is already used by another VovServer or by another application. VOV computes the port number in the range [6200, 6455] by hashing the project name. If necessary, select another project name, or change host, or use the variable `VOV_PORT_NUMBER` to specify a known unused port number. The best place to set this variable is in the `setup.tcl` file for the project.

- Check if the server is trying to use an inactive port number that cannot be bound. This can happen when an application, perhaps the server itself, terminates without closing all its sockets.

  The server will exit with a message similar to the following:

```
...more output from vovserver...
vs52 Nov 02 17:34:55          0                3      /home/john/vov
vs52 Nov 02 17:34:55 Adding licadm@venus to notification manager
vs52 Nov 02 17:34:55 Socket address 6437 (net=6437)
vs52 ERROR Nov 02 17:34:55 Binding TCP socket: retrying 3
vs52 Nov 02 17:34:55 Forcing reuse...
vs52 ERROR Nov 02 17:34:58 Binding TCP socket: retrying 2
vs52 Nov 02 17:34:58 Forcing reuse...
vs52 ERROR Nov 02 17:35:01 Binding TCP socket: retrying 1
vs52 Nov 02 17:35:01 Forcing reuse...
vs52 ERROR Nov 02 17:35:04 Binding TCP socket: retrying 0
vs52 Nov 02 17:35:04 Forcing reuse...
vs52 ERROR Nov 02 17:35:04
PROBLEM:  The TCP/IP port with address   6437   is already being used.

POSSIBLE EXPLANATION:
        - A VOV server is already running   (please check)
        - The old server is dead but some
          of its old clients are still alive  (common)
        - Another application is using the
          address  (unlikely)

ACTION: Do you want to force the reuse of the address?
```

  In this case:

  1. List all VOV processes that may be running on the server host and that may still be using the port. For example, you can use:

```
% /usr/ucb/ps auxww | grep vov
john   3732  0.2  1.5 2340 1876 pts/13   S 17:36:18  0:00 vovproxy -p acprose -f - -b
john   3727  0.1  2.2 4816 2752 pts/13   S 17:36:16  0:01 vovsh -t /opt/rtda/latest/linux64/tcl/vtcl/vovresourced.tc
...
```

  2. You can wait for the process to die on its own,  or you can kill it, for example with `vovkill`

```
% vovkill pid
```

  3. Restart the server.

- You run the server as the WorkloadXelerator administrator user. Please check the ownership of the file `security.tcl` in the server configuration directory `vwx.swd`.

### 2.10.2 The Unix Slaves Do Not Start

WorkloadXelerator normally relies on remote shell execution to start the slaves using either `rsh` or `ssh`.

- If using `rsh` try the following:

```
% rsh host vovarch
```

  where *host* is the name of a machine on which there are problems starting a slave.

  This command should return a platform dependent string (such as "linux") and nothing else. Otherwise, there are problems with either the remote execution permission or the shell start-up script.
- If the error message is similar to "Permission denied", check the file `.rhosts` in your home directory. The file should contain a list of host names from which remote execution is allowed. See the manual pages for `rsh` and `rhosts` for details. You may have to work with your system administrators to find out if your network configuration allows remote execution.
- If using `ssh`, perform the test above but use `ssh` instead of `rsh`. For more details about `ssh` check the section SSH Setup  in the VOV™ Subsystem Administrator Guide.
- If you get extraneous output from the above command, the problem is probably in your shell start-up script. If you are a C-shell user, check your `~/.cshrc` file. Following are guidelines for a remote-execution-friendly `.cshrc` file:
  - Echo messages only if the calling shell is interactive. You can test if a shell is interactive by checking the existence of the variable prompt, which is defined for interactive shells. Example:

```
# Fragment of .cshrc file.
if ( $?prompt ) then
echo "I am interactive"
endif
```

- Many `.cshrc` scripts exit early if they detect a non-interactive shell. It is possible that the scripts exit before sourcing `~/.vovrc`, which causes WorkloadXelerator to not be available in non-interactive shells. Compare the following fragments of `.cshrc` files and make sure the code in your file works properly:

  The following example will not work properly for non-interactive shells:

  ```
  if ( $?prompt ) exit
  source ~/.vovrc
  ```

  This example is correct; source .vovrc and then check the prompt variable:

  ```
  source ~/.vovrc
  if ( $?prompt ) exit
  ```

  This example is also correct:

  ```
  if ( $?prompt ) then
  # Define shell aliases
  ...
  endif
  source ~/.vovrc
  ```

- Do not apply `exec` to a sub-shell. This will cause the `rsh` command to hang.

  ```
  # Do not do this in a .cshrc file
  exec tcsh
  ```

### 2.10.3 License Violation

WorkloadXelerator is licensed by restricting the number of slave slots. This is the sum of the slave slots from both elastic and statically defined slaves (if any).

You can find out the capacity of your license (wx_slots) with the following command:

```
% rlmstat -avail
```

The file `$VOVDIR/../../vnc/vwx.swd/slaves.tcl` defines the list of static slaves that are managed by the server. Make sure the number of slave hosts is within the license capability.

### 2.10.4 Crash Recovery

In the event of a crash or failover, you can find a checklist of what to do at http://wx-host:wx-port/cgi/sysrecovery.cgi.
This address can found using the command:

```
wx cmd vovbrowser -url /cgi/sysrecovery.cgi
```

## 2.11 Basic WX troubleshooting

### 2.11.1 When Things Go Wrong

The following may occur after a major infrastructure event, a problem with submission scripts or a change to the workload or a change to things like Limits.

The most significant symptom is "stuck jobs": most likely `elastic daemon` has stopped. Check the `elastic daemon` log
**Note:** Both cases are common NetworkComputer debug scenarios.

**If the timestamp is not fresh, you will need to restart:**

1. Save off the existing log file (and send that file to RTDA for diagnosis).
2. Restart: `nc -f $WxQueueName cmd vovautostart`

**If elastic daemon is running:**

1. Check that slave jobs are being submitted and that they are being executed by the base scheduler. To do this, connect to the correct NC cluster through the web browser, locate the job set called *vovelasticd*. This set contains other sets, one set for each WorkloadXelerator session.
2. Locate the appropriate set for your WorkloadXelerator session. Look at the name of the set.
3. If you see only cyan (scheduled) jobs, the problem is that the base scheduler cannot schedule these slave jobs. You need to debug why these jobs are not being run by the underlying scheduler.
4. If the jobs are getting run but keep failing (turning red) then debug and determine the reason for those failures.

### 2.11.2 When Jobs Are Not Running

**Resource Issues**

You may discover that a slave job is asking for an impossible resource. This is often due to an error in the resource requirements, or an error in the configuration. You fix the problem but WorkloadXelerator continues to not run jobs. In this case, those "impossible" jobs are still seen by the base scheduler (and are therefore not runnable) but also they are seen by the elastic daemon, which assumes that these jobs are runnable and that no new jobs should be submitted: a *live-lock* scenario.

The recommendation here is to dequeue any queued jobs in the base scheduler after changing the problematic resource request. This lets the elastic daemon launch replacement jobs.

A different scenario is slave jobs being scheduled, dispatched and running in the base but no jobs are getting executed inside the WorkloadXelerator session. The symptom is a growing list of green nodes (valid) with a small number of orange (running) and cyan (scheduled) jobs. In this case, the base is dispatching the slave jobs without a problem, but WorkloadXelerator is not making use of them. The results: they execute, waiting for the end user job that never comes until they hit their maximum idle limit.

For this case, we recommend checking the WorkloadXelerator session using either a vovconsole (wx -q $WxQueueName cmd vovconsole) or a WX monitor. If you see no activity in the LED bar (slaves connecting, waiting and then terminating - often yellow-to-green-to-black), most likely there is a problem with the configuration. The slaves that the base scheduler is executing are not connecting back to the desired WorkloadXelerator session.

For the configuration, verify that the slaves are being launched with the right parameters and check that the config files are correct:

- If the LED monitor is active, then slaves are connecting and the failure to launch is mostly likely in WorkloadXelerator. A common problem is that the job requests a limit or a special resource: the limit must be satisfied in WorkloadXelerator and the base scheduler. In this case, elastic daemon tries to detect expandable limits sucy as Limit:foo_@USER@_N and will set the limit to be unlimited in WorkloadXelerator, and then pass the limit request on to the base scheduler where it is honored. Occasionally, this process goes wrong.
- If the limit does not exist or is set to 0 in WorkloadXelerator, then the job will not launch. In this state, the job will appear to be queued (cyan color), but you will not see a bucket created for it (wx mon can be used to help diagnose that case). To add the missing resources edit `resources.tcl`. In general, set the resource value to "unlimited" in WorkloadXelerator; the restrictive value in the base scheduler will still be honored.

### 2.11.3 What do Check When Jobs are Not Being Submitted

WorkloadXelerator reports the slave as SICK and it has a missing heartbeat. Check in the base scheduler and see that the slave has been suspended. In this case someone or something has suspended the slave - both it and it's underlying job are suspended (T-state/CTRL-Z). Find out why - it could be a user or RAM sentry - the slave job may have some property information telling you what. When resumed the slave will become healthy (i.e. non SICK) and continue normally within the base scheduler and WorkloadXelerator.

Everything seems fine but jobs are not being submitted. Check the base scheduler and if you find that the slave jobs aren't running due to fair share reasons then it may be possible there's a regression that is using the same fairshare node and subgroup (either from WorkloadXelerator or native in the base scheduler). In this case the jobs will be treated in a first-come-first-served basis.

### 2.11.4 Avoid Suspending WorkloadXelerator slaves in the base scheduler

**Note:** elastic slave jobs in the base scheduler should not be subjected to suspension events from users or the preemption system. WorkloadXelerator supports a form of preemption called **modulation.**. In this case, the base scheduler queue requests the WorkloadXelerator slave to terminate on the next WorkloadXelerator job boundary.

## 2.12 Job Placement Policies

One of the few differences between WX and NC is that, while NC supports many placement policies, WX only supports the job placement policy called "first", meaning that a job is dispatched to the first slave found that is capable of running the job.

You can still specify any JPP as you submit the workload, but that JPP is passed to NC (assuming NC is the base scheduler) while WX itself uses the "first" JPP.

For a complete description of job placement policies, look at the NC documentation on JPP

## 2.13 Web Interface

Some features of the WorkloadXelerator web interface can be configured by the administrator. Configuration for these items is performed in the `vnc.swd/config/web.cfg` file. The complete list of customizable items is shown below.

**Example Configuration: $VOVDIR/etc/config/nc/web.cfg**

```
### NODE VIEWER SETTINGS ###

# Use a select (drop-down) menu for priority-based retrace controls
# Set to 0 to disable, set to 1 or comment-out to enable (default)
# set nodeviewer(retraceSelect) 1

# Use a select (drop-down) menu for preemption controls
# Set to 0 to disable, set to 1 or comment-out to enable (default)
# set nodeviewer(preemptSelect) 1
```

## 2.14 Using NetworkComputer as a base scheduler

When interfacing to NC, we recommend a minimal (zero) configuration for WorkloadXelerator; do not configure each WorkloadXelerator instance (resources, limits, etc.), and instead rely on the base scheduler's configuration to implement these policies. This approach allows many WorkloadXelerator instances to be employed without a corresponding increase in administration overhead.

A notable exception is with queue-specific job classes: job classes need to be present in the WorkloadXelerator instance otherwise an error is flagged. To avoid duplication of code, it may be desirable to symlink the base scheduler's jobclass directory to the WorkloadXelerator instance. It may be necessary to copy or symlink NetworkComputer's vnc_policy.tcl file to the WorkloadXelerator instance, for example: data specific to the job submission environment is required to run the policies.

Also, the jobclasses must be written so that they can be sourced multiple times, (e.g. to prevent duplicate entries in the resource lists), since it is likely that they will be executed once in the WX context and once in the NC context.

The use of preemption within WorkloadXelerator is not recommended; it is best to let NC do all necessary preemptions, which will result in job modulation.

FairShare works the same in WX and in NC. You will notice that, in NC, fairshare convergence will be somewhat slower than you would expect, because WX is effectively sending to NC much longer jobs than the jobs in the WX workload, due to the fact that NC sees the "vovslave" as a single job.

## 2.15 Many WXs on many NCs

For "many to many" model, where multiple WX queues are running over multiple NC queues, the user should create separate WX queues, and configure the vovwxd daemon for each of them, then specify `CONFIG(queues)` parameter accordingly for each WX queue.

# 3 WorkloadXelerator vovslave Configuration

## 3.1 Black Hole Detection in WX

For an introduction at what a black-hole slave is, please refer to the Section on Black Hole Detection

Since the slaves in WX are all coming from an underlying base scheduler, it makes sense to assume that such slaves are considered

If black-hole detection is activated, and a slave is determined to be a black-hole, it is immediately stopped. In NC, instead, a black-hole slave is temporarily suspended and then is allowed to receive jobs after the suspension.

# 4 Job Classes

## 4.1 Job Class Administration

### 4.1.1 Introduction to Job Classes

Job classes provide the following advantages:

1. Simplifying the command line for job submission, which can prevent errors and omissions.
2. Emulate the concept of *queues*, which is similar to the processes of other batch processing systems. This queue emulation enables additional behaviors such as:
    - Automatic revocation of resources that have been grabbed by jobs in the jobclass but are not used
    - Automatic warning and termination of jobs that are stuck: jobs that have been dispatched to a vovslave but appear to be using no CPU time

A jobclass represents a collection of *wx run* options that are needed to run a type of jobs, such as VCS regression jobs.

Membership in a jobclass can be used to differentiate between jobs in preemption: preempt jobs in a *regression* jobclass to free up resources for jobs in an *interactive* jobclass.

If more than one `-C` option is given, the jobclasses are processed left-to-right as the command line is parsed. This method requires great care and planning.

### 4.1.2 Creating Job Classes

The administrator of WorkloadXelerator can define jobclasses using one of the following methods:

1. Logged in as ADMIN, click the **Job classes** link in the **Workload** section of the WorkloadXelerator main page. This page displays all of the available jobclasses, and allows creating and editing jobclasses, and allows the administrator to designate a default jobclass.
2. Directly add Tcl-syntax files in the directory `jobclass` under the server working directory, which is typically normally `$VOVDIR/../../vnc/vnc.swd/jobclass`.

Each file in the `jobclass` directory manipulates the submission parameters defined in the Tcl array VOV_JOB_DESC so as to define a jobclass. See the following table for the meanings of the items in this data structure. The variable `classDescription` holds a string used for documentation, i.e. a one-line summary of the jobclass.

An optional procedure `initJobClass` can be defined to do any initializations needed for the jobclass to perform correctly. Often, this is used to create any Limit: resources that may be used by the jobclass.

The files in the jobclass directory are parsed by `vovresourced` when it starts, and any `initJobClass` procedures are evaluated once.
**IMPORTANT**: Whenever you change an `initJobClass` procedure, you must restart `vovresourced` to put the change into effect.

The jobclass definition files are located using a search path.

The built-in search path is computed by a procedure `VncJobClassSearchPath`, and is shown in the table below.

## 4.2 Reconciling Unused Resources

The use of license revocation is not recommended with WorkloadXelerator.

## 4.3 Defining job classes

This `VOV_JOB_DESC` data structure is an associative array that describes the characteristics of a job. The array has a number of slots that hold the values describing the job. The following table shows the the array slot fields.

| Field in array | Description |
|---|---|
| **autokill** | Set the autokill flag (option `-kill`) |
| **check,directory** | Set it to 0 to disable checking of canonicalization of current directory (option -D) |
| **env** | Environment of the job (option `-e`). Set this to "" or to *DEFAULT* to force the use of an environment snapshot. |
| **force** | Force the job to be rescheduled (option `-F`) |
| **group** | Group the job belongs to (options `-g` and `-G`) |
| **inputs** | List of input files (dependencies) (option `-i`) |
| **interactive,flag** | Used for interactive jobs, with values `tty_remote` (option `-Ir`) or `tty_local` (option `-Il`) |
| **interactive, useXdisplay** | Set if the job requires an X display (option `-Ix`) |
| **logfile** | Name of the log file (option `-l`) |
| **mailuser** | Specification of who gets the e-mail notification (option `-M`) |
| **osgroup** | The Unix group this job. This field is read-only and cannot be changed (see also *user*) |
| **outputs** | List of output files (option `-o`) |
| **preemptable** | A suggestion to determine if the job is preemptable |
| **priority, default** | Default priority (NOT USED) |
| **priority, exec** | Execution priority |
| **priority, sched** | Scheduling priority |
| **proplist** | Properties to be added to the job (option `-P`) |
| **resources** | The resources of the job (option `-r`) |
| **rundir** | The running directory for the job (normally .) |
| **schedule, date** | NOT SUPPORTED YET |
| **setName** | The set to which the job belongs (option `-set`) |
| **user** | The user for this job. This field is read-only and cannot be changed (see also *osgroup*) |
| **wait** | Boolean: set it to `1` to wait for the job to complete (option `-w`) |
| **wait,options** | When waiting, these options are passed to the `nc wait` command. For example, set it to `-l` to view the log file |
| **wrapper** | Wrapper used for the job (option `-wrapper`) |
| **xdur** | Expected duration of the job (option `-X`) |

Here is an example of the filled-in `VOV_JOB_DESC` array.

**Example output of 'parray VOV_JOB_DESC'**

```
VOV_JOB_DESC(autoforget)              = 0
VOV_JOB_DESC(autokill)                = 1
VOV_JOB_DESC(check,directory)         = 1
VOV_JOB_DESC(env)                     = BASE+RTSIM
VOV_JOB_DESC(force)                   = 0
VOV_JOB_DESC(group)                   = users
VOV_JOB_DESC(inputs)                  =
VOV_JOB_DESC(interactive,flag)        = none
VOV_JOB_DESC(interactive,useXdisplay) = 0
VOV_JOB_DESC(logfile)                 = vnc_logs/20050920/131409.25563
VOV_JOB_DESC(mailuser)                =
VOV_JOB_DESC(osgroup)                 = guests
VOV_JOB_DESC(outputs)                 =
VOV_JOB_DESC(priority,default)        = 4
VOV_JOB_DESC(priority,exec)           = 4
VOV_JOB_DESC(priority,sched)          = 8
VOV_JOB_DESC(proplist)                =
VOV_JOB_DESC(resources)               = linux
VOV_JOB_DESC(rundir)                  = .
VOV_JOB_DESC(schedule,date)           = 0
VOV_JOB_DESC(setName)                 =
VOV_JOB_DESC(user)                    = mary
VOV_JOB_DESC(wait)                    = 0
VOV_JOB_DESC(wait,options)            =
VOV_JOB_DESC(wrapper)                 = vw
VOV_JOB_DESC(xdur)                    = 30
```

## 4.4 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.

### 4.4.1 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.

### 4.4.2 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.

# 5 Resources and Licenses

## 5.1 Resource Management

VOV includes a subsystem for managing computing resources. This allows the design team to factor in various constraints regarding hardware and software resources, as well as site policy constraints. This mechanism is based on the following:

- Resources required by jobs
- Resources offered by slaves
- Resource Maps, as described in the file `resources.tcl`

There are several types of resources, which are listed below:

| Resource type | Representation | Explanation |
|---|---|---|
| Job Resources | `name` | A resource required by a job. If the quantity is not shown, the default is 1; "unix" is the same as "unix#1". |
| Uncountable Resources (also called Attributes) | `name` | These resources represent attributes of a slave that are not countable. For example, a slave may have attributes such as "unix" or "linux". The quantity is not shown for these resources and it defaults to MAXINT; "unix" is equivalent to "unix#MAXINT". |
| Quantitative Resources | `name#quantity` | Example: The resource `RAMTOTAL#2014` on a slave indicates the total amount of RAM on that machine. On a job, it says that the job requires **at least** the shown amount of RAMTOTAL. |
| Consumable Resources | `name/quantity` | Example: RAM/500 assigned to a job indicates that the job consumes 500 MB of the consumable resources RAM. |
| Negated Resources | `!name` | Example: "unix !linux" on a job indicates that the job requires a UNIX machine but not a Linux one. |

The definition of the quantity is related to the context of the resource. If the context is a slave , quantity represents how much of that resource is available from the slave. If the context is a job, quality represents how much of that resource is required by the job.

**Note:** Negated resources are allowed only for the context of a job.

The *unit of measure* is determined by convention for each resource. For example, the resource `RAMTOTAL` is measured in MB. By default, quantity is assumed to be 1; the notation `foo` is equivalent to `foo#1`.

A *resources list* is a space-separated list of resources, which are typical resources offered by the slaves. The following example indicates that a job requires at least 128 MB of RAM and a Unix host, but not a linux host.

```
RAMTOTAL#128 unix !linux
```

A *resources expression* is a space separated list of resources and operators: typical resources requested by the jobs or mapped in the resource map set. Operators can be one of the following: `<blank space>`, `&`, `|`, `OR`, `AND`, `!`, and `NOT`. The operators are defined in the table below. **Note:** Logical `AND` has precedence over logical `OR` operations.

| Operator | Description |
|---|---|
| `<blank space>` | implicit logical AND |
| `&` | explicit logical AND |
| `AND` | explicit logical AND |
| `\|` | explicit logical OR |
| `OR` | explicit logical OR |
| `!` | explicit logical negation |
| `NOT` | explicit logical negation |

For example, a job may have the following resource requirements:

```
RAMTOTAL#128 unix !linux | RAMTOTAL#512 & linux
```

This job requires either a unix host with at least 128 MB of RAM, but not a `linux` host or a linux host with at least 512MB of RAM.

## 5.2 Requesting Hardware Resources

All slaves offer a predefined set of hardware resources that can be requested by jobs. These resources are listed in the following table.

| Hardware Resource | Type | Description |
|---|---|---|
| ARCH | STRING | The VOV architecture of the machine, for example "linux", "win64", "armv7l" |
| CORES | INTEGER | The total number of cores in the machine. On Linux, this is the value returned by `sysconf(_SC_NPROCESSORS_ONLN)`. |
| CORESUSED | INTEGER | The total number of cores used by the running jobs. It is assumed that each job uses at least one core. |
| CLOCK | INTEGER | The CPU-clock of at least one of the CPUs on the machine in MHz. If the machine allows frequency stepping, this number can be smaller than expected. |
| GROUP | STRING | The slave group for this slave. Each slave can belong to only one slave group. |
| HOST | STRING | The name of the host on which the slave is running. Typically this is the value you get with `uname -n`, except we take only the first component and we convert it to lowercase, so that if `uname -n` returns `Lnx0123.my.company.com` the value of this field will be `lnx0123`. |
| LOADEFF | REAL | The effective load on the machine, including the self-induced load caused by jobs that just started or finished. |
| L1 | REAL | On Unix, the load average in the last one minute. |
| L5 | REAL | On Unix, the load average in the last five minutes. |
| L15 | REAL | On Unix, the load average in the last fifteen minutes. |
| MACHINE | STRING | Typically the output of `uname -m`. |
| NAME | STRING | The name of the slave. |
| OS | STRING | The name of the operating system. |
| OSCLASS | STRING | This can be `unix` or `windows`. |
| OSVERSION | STRING | Typically the output of `uname -v`. |
| OSRELEASE | STRING | Typically the output of `uname -r`. |
| PERCENT | INTEGER | The percentage of the machine that is still available. |
| POWER | INTEGER | The effective power of the slave, after accounting for both raw power and the effective load. |
| RAM | INTEGER | A consumable resource expressing the remaining RAM available to run job: RAMTOTAL-RAMUSED, in MB. |
| RAMFREE | INTEGER | The amount of RAM available to run other jobs. This metric comes from the OS, and on linux it includes both free memory and buffers. In MB. |
| RAMTOTAL | INTEGER | The total amount of RAM available on the machine, in MB. |
| RAMUSED | INTEGER | The cumulative amount of RAM used by the jobs currently running on the slave, in MB. |
| RELEASE | STRING | On Linux machines, this is the output of `lsb_release -isr`, with spaces replaced by dashes. For example, `CentOS-6.2` |
| SLAVENAME | STRING | Same as *NAME* |
| SLAVEHOST | STRING | Same as *HOST* |
| SLOT | INTEGER | A consumable resource indicating how many more jobs can be run on the slave. |
| SLOTS | INTEGER | Same as SLOT |
| SLOTSUSED | INTEGER | Corresponding to the number of jobs running on the slave. |
| STATUS | ENUMERATED TYPE | Possible values are `BLACKHOLE BUSY DEAD DONE FULL OVRLD NOLIC NOSLOT OK PAUSED READY REQUESTED SICK SUSP WARN WRKNG` |
| SWAP | INTEGER | Consumable resource |
| SWAPFREE | INTEGER | The amount of free swap. |
| SWAPTOTAL | INTEGER | Total about of swap configured on the machine. |
| TIMELEFT | INTEGER | The number of seconds before the slave is expected to exit or to suspend. This value is always checked against the expected duration of a job. |
| TMP | INTEGER | On unix, free disk space in /tmp, in MB. |
| USER | STRING | The user that is running the slave. Normally this user is *root* on Unix. |
| VOVVERSION | STRING | The version of the vovslave binary (such as '2015.03'). |

### 5.2.1 Requesting Hardware Resources

Each job can request hardware resources.
**Note:** The consumable resources are `CORES`, `CPUS`, `RAM`, `SLOT`, `SLOTS`, and `SWAP`.

- To request a machine with the name *bison*, request `NAME=bison`. To request any linux64 machine, request `ARCH=linux64`.
- Consumable resources are added together. For example `-r CORES/2 CORES/4 CORES/6` is a request for a total of 12 cores.
- If redundant resources are specified, the largest value will be taken. For example, if `-r RAM#2000 RAM#4000` is specifiec, then `RAM#4000` will be the resource that is used.

Request examples are listed in the following table:

| Request Objective | Syntax for the Request |
|---|---|
| A specific slave | `NAME=bison` |
| Not on bison | `NAME!=bison` |
| One of two slaves | `NAME=bison,cheetah` |
| A preference: bison, if it is available; otherwise, cheetah | `(NAME=bison OR NAME=cheetah)` |
| A specific architecture, such as linux | `ARCH=linux` |
| A specific slave group, such as prodLnx | `GROUP=prodLnx` |
| 2 GB of RAM | `RAM/2000` |
| Two cores | `CORES/2` |
| Two slots | `SLOTS/2` |
| Exclusive access to a machine | `PERCENT/100` |
| 1 minute load less than 3.0 | `L1<3.0` |

## 5.3 Resource Mapping

As the VovServer determines which slave is most suited to execute a particular job, it performs a *mapping* of the job resources, followed by a *matching* of the mapped resources.

When dispatching a job, the VovServer does the following:

1. Gets the list of resources required by a job.
2. Appends the resource associated with the priority level, such as `Priority:normal`.;
3. If it exists, it appends the resource associated with the name of the tool used in the job (reminder: the tool of a command is the tail of the first command argument after the wrappers). The tool resource has type "Tool" and looks like this: `Tool:`*`toolname`*.
4. Appends the resource associated with the owner of the job, such as `User:john`
5. Appends the resource associated with the group of the job, such as `Group:time_regression`.
6. Expands any special resource, i.e. any resource that starts with a "$";
7. For each resource in the list, the VOV server looks for it in the Resource Maps. If the resource map is found and there is enough of it, i.e. the resource is available, the VOV server maps the resource. This step is repeated until one of the following conditions is met:
    - The resource is not available. In this case, the job cannot be dispatched and is left in the job queue.
    - A cycle in the mapping is detected; in this case the job cannot be dispatched at all and is removed from the job queue.
    - The resource is not in the Resource Map.
8. VOV appends the resource associated with the expected job duration to the final resource list.. For example, if the job is expected to take 32 seconds, the resource `TIMELEFT#32` will be appended.
9. Finally, the VovServer compares the resulting resource list with the resource list of each slave. If there is a match - all resources in the list are offered by the slave - the slave is labeled as eligible. If there is no eligible slave, the job cannot be dispatched at this time and remains in the queue; otherwise, the server selects the eligible slave with the greatest effective power.

## 5.4 Automatic Resource Limits

The resources of type "Limit" are treated specially by VOV. When a job is created or submitted, if the name of the job resource contains one or more of these tokens
@USER@ @GROUP@ @JOBCLASS@ @JOBPROJ@
then each token is replaced by the value of the corresponding field for the job. The resourcemap with the token is called the **symbolic limit** while the derived resourcemap is called the **specific limit**

For example, if user "john" submits a job with the resources "Limit:abc_@USER@", the following happens:

- The resource requirement for the job is changed so that Limit:abc_@USER@ is replaced with Limit:abc_john
- A new resource map called Limit:abc_john is created. This resource map will be assigned a maximum amount equal to the maximum of the resourcemap called "Limit:abc_@USER@", if such resourcemap exists, or just 1 if the resource does not exist.

To use the automatic limit resources, the admin needs to create the symbolic resource. For example:

```
# In resources.tcl
vtk_resourcemap_set Limit:queue_normal_@USER@ 10
```

To change the value of all limits derived from a symbolic limit, you should use the procedure `vtk_resourcemap_set_limit`. This procedure normally sets all derived limits to the same new value, but it also allows the specification of different limits for selected users or the reduction of specific limits based on the out-of-queue usage of a given license.

```
# Example 1: set all derived limits to 15:
vtk_resourcemap_set_limit Limit:queue_normal_@USER@ 15

# Example 2: set all derived limits to 15, with a few exceptions:
vtk_resourcemap_set_limit Limit:queue_normal_@USER@ 15 -special {
   Limit:queue_normal_mary 20
   Limit:queue_normal_john  3
}

# Example 3: set all derived limits to 15, but consider out-of-queue usage
#            Since this limit changes over time, we put it inside a TIMEVAR
#            procedure, so it is computed once every minute.
TIMEVAR hsim_ooq {
   default {
       vtk_resourcemap_set_limit Limit:queue_hsim_@USER@ 15 -ooq License:hsim
   }
}
```

# 6 Frequently Asked Questions and Troubleshooting Tips

## 6.1 HPC Advice

This section provides recommendations to obtain the maximum performance from WorkloadXelerator.

### 6.1.1 Use the Latest Runtime Release

The performance of the WorkloadXelerator scheduler is frequently updated. Using the most current version is recommended.

### 6.1.2 Use the vwn wrapper

The wrapper `vwn` (alias for `vw -N`) avoids communication with vovserver. An example is shown below:

```
% wx run -wrapper vwn sleep 0
```

- The benefit is of using `vwn` is speed.
- The disadvantage is that jobs that require the `-wl` option cannot be run. However, this disadvantage may be not be significant, as `-wl` adds a relatively high load for what it does: `-wl` requires an extra *notify client* to handle the event generated when the job terminates.

### 6.1.3 Disable Wait Reasons

If analyzing what causes *wait time* in the workload, the wait reason analisys can be disabled as shown below:

```
# In policy.tcl
set config(enableWaitReasons) 0
```

Wait time analysis can then be re-enable as needed as shown below:

```
% wx cmd vovsh -x 'vtk_server_config enableWaitReasons 1'

### collect some data for a few minutes, then

% wx cmd vovsh -x 'vtk_server_config enableWaitReasons 0'
```

### 6.1.4 Disable File Access

Disabling file access is mostly a high-reliability option. By disabling file access, the VovServer never looks at any of the files in the user workspaces, which avoids the risk of disk slowness or disk unavailability. An example is shown below:

```
% wx cmd vovsh -x 'vtk_server_config disablefileaccess 2'
```

### 6.1.5 Reduce Update Rate of Notify Clients

*Notify clients*, clients that are tapping the event stream from VovServer (such as `wx gui`, `voveventmon` or `wx run -wl`), are updated immediately in the inner loop of the scheduler. If the environment includes hundreds of such clients, it may be beneficial to slow down the update rate by setting the parameter `notifySkip`. The default value is 0: no skip. Typically,the more events that take place, the more events that can be skipped without notice. For example, if several events are taking place, setting `notifySkip` to 100, fewer updates may not be noticed. If the number of events is small, a one-second delay may be noticed in some updates of the GUI. skipped without notice.
**Note:** Regardless of the setting, the maximum time between updates is one second.

```
# In policy.tcl
set config(notifySkip) 100
```

## 6.2 WX Modulation

When using WorkloadXelerator, jobs launched in WX are essentially bundled into groups that are run by vovslave on hosts allocated by the base scheduler. This means that it is harder to depend on job retirement to free up slots in the base scheduler, because the bundle of jobs is of course many times longer than the individual jobs.

This section describes a means of freeing up slots more quickly by preempting the vovslaves that have been assigned to WX, based on fairshare statistics. A preempted vovslave will stop accepting jobs (slave status "DONE") and will still finish any running job.

The preemption rule drives the system and this is the main place to influence the systems behavior. A sample rule is found in $THISGIT/vovpreempt/config.tcl and this should be appended to any existing preemption rules in XXXX.swd/vovpreemptd/config.tcl.

While the rule can be tuned there are some key elements that must be retained. Preemptable jobs should be have the predicate JOBNAME~${WXQueueName} and the method should send SIGUSR2 but only to the vovslave process: "0:*:EXT,SIGUSR2,vovslave".

The preemptable job sort predicate is "FS_EXCESS_RUNNING DESC, PRIORITY, AGE DESC" which chooses vovslaves ordered on greatest excess fairshare, lowest priority and oldest age.

Here is an example of a preemption rule for job modulation in WX:

```
# Taken from $VOVDIR/etc/config/vovpreemptd/config_wx_modulation.tcl
set WXQueueName wx
VovPreemptRule \
    -pool     "WXJobModulation" \
    -rulename "fastFairshare_$WXQueueName" \
    -ruletype "FAST_FAIRSHARE" \
    -method   "0:*:EXT,SIGUSR2,vovslave" \
    -killage   0 \
    -numjobs  10 \
    -maxattempts 1 \
    -waitingfor "HW" \
    -preempting  "JOBNAME~${WXQueueName} FS_EXCESS_RUNNING<0" \
    -preemptable "JOBNAME~${WXQueueName} FS_EXCESS_RUNNING>0 FSRANK9&gt>@FSRANK9@" \
    -resumeres "" \
    -enable     1 \
    -sortjobs "FS_EXCESS_RUNNING DESC,PRIORITY,AGE DESC"
```

### 6.2.1 Monitoring

This is a dynamic system with quite a few moving parts and this makes monitoring a bit challenging. Some suggestions follow.

- Turn on the debug option in the preemption rule - the preemption activity will be logged in a property attached to the preemption rule object. Use the global preemption debug flag to get the info also in the main vovserver log.

```
% vovsh -x 'vtk_server_config   set_debug_flag PreemptRules'


% vovsh -x 'vtk_server_config reset_debug_flag PreemptRules'
```

## 6.3 Preemption of WX Agents

### 6.3.1 Preempting WX agents from a NC queue using NC preemption

vovslave understands Signal USR1 as killing all the jobs running on it and exit as soon as possible. This signal can be used to preempt WX agents running on a NC queue.
When WX agent receives a USR1, it kills all jobs, set the states of jobs as WITHDRAWN. These jobs get rescheduled in WX and will be dispatched on a new slave. WX agent on NC exits and becomes a valid job.
The following is an example to preempt a WX agent job on NC. `nc preempt -method "0:*:SIGUSR1" 159500`

# 7 Legal

## 7.1 Runtime Inc Copyright



### 7.1.1 ArgumentParser

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



### 7.1.2 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.

---

### 7.1.3 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.

---

### 7.1.4 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*.

---

### 7.1.5 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).

### 7.1.6 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.

### 7.1.7 [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.

### 7.1.8 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.

### 7.1.9 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.

---

### 7.1.10 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.

---

### 7.1.11 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.

---

### 7.1.12 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.



---

### 7.1.13 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.

---

### 7.1.14 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.

### 7.1.15 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.

### 7.1.16 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.

### 7.1.17 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.

### 7.1.18 SQLite

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

### 7.1.19 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.

### 7.1.20 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.

### 7.1.21 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.

### 7.1.22 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

### 7.1.23 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.

### 7.1.24 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).

### 7.1.25 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).