



ALTAIR

ONLY FORWARD

Altair Monitor 2025.1.2

Administrator Guide

Contents

Altair Monitor Overview	4
System Description	5
Enable CLI Access on UNIX	9
Troubleshooting the UNIX Setup	12
Enable CLI Access on Windows	13
Verify Context Is Working	14
Use Monitor Help	16
Server Operation	18
Troubleshooting: Images Do Not Display	23
Failover Server Candidates	25
Crash Recovery Mode	28
Daemon Operation	29
Daemon Manager Utility	30
Database	32
Daemon	32
Tasker	32
Set Up	32
Configure the Database from the Command Line	34
Database Engine Versions and Upgrades	37
Load Data	38
Export Data	40
Database Backup	42
Tag Administration	43
Manage Data from Different Origins	46
License Violation Effects and Actions	46
Database Schema	49
Configuring Monitoring	50
General Monitoring Configuration	50
License Server Monitoring	54
Debug Log Monitoring	58
False Denial Filtering	65
Multi-Site Support	66
Project and Group Management	68
Network Monitoring	78
Monitor Processes	84
Wrapping Unlicensed Tools	85
License Server Management	89
License Management Overview	89
Get Started with ControlCenter	89
Agent Configuration	95
Command Line Setup	96

Other Configurations.....	98
Notification.....	98
Security.....	102
Web Server Configuration.....	107
Web Interface.....	111
LDAP Integration.....	114
Workweek Definition.....	118
Command Line Operations.....	119
Sample a License Server.....	119
Parse a Debug Log.....	120
Load the Database.....	120
Set Feature Capacity.....	121
Update Summary Table.....	122
VovUserGroups.....	123
Query the vovserver.....	125
Reference.....	133
Troubleshooting.....	133
Troubleshooting: Images Do Not Display.....	137
Performance Tuning.....	137
Data Files.....	138
Configuration Commands.....	140
Autostart Directory.....	169
Data Collection Commands.....	170
Database Loading Commands.....	184
Maintenance Tasks.....	185
Accuracy.....	186
Access Control List.....	187
Widgets.....	191
Time Zones.....	213
Legal Notices	215
Intellectual Property Rights Notice.....	216
Technical Support.....	222
Index	223

This document describes how to configure and manage Altair Monitor.

This chapter covers the following:

- [System Description](#) (p. 5)
- [Enable CLI Access on UNIX](#) (p. 9)
- [Enable CLI Access on Windows](#) (p. 13)
- [Use Monitor Help](#) (p. 16)
- [Server Operation](#) (p. 18)
- [Failover Server Candidates](#) (p. 25)
- [Daemon Operation](#) (p. 29)
- [Daemon Manager Utility](#) (p. 30)
- [Database](#) (p. 32)
- [Configuring Monitoring](#) (p. 50)
- [License Server Management](#) (p. 89)
- [Other Configurations](#) (p. 98)
- [Command Line Operations](#) (p. 119)
- [Reference](#) (p. 133)

You must have already installed the software by following the instructions and checklists in the *Installation Guide*.

The following chapters describe how to:

- Configure monitoring of software utilization (licensed and unlicensed) and of hardware and operating systems.
- Start and stop Monitor.
- Configure management of license servers and files.
- Manage the Monitor database.

System Description

Monitor provides easy and comprehensive monitoring of software licenses and computing hardware. This section gives an overview of Monitor's functionality.

Features

- **Web Interface:** The main interface for viewing information gathered by Monitor is a browser interface viewable on any web browser that supports JavaScript, CSS, and AJAX. Generally, Monitor works with any modern browser.
- **Centralized Monitoring of Licenses:** Monitor supports monitoring of FlexNet Publisher licenses, as well as other license managers. For software that does not utilize license manager software, Monitor can be used to provide basic licensing capabilities.
- **Database Driven:** A high-performance, highly-scalable relational database server is fully integrated and managed by Monitor. The database is SQL-based, allowing for complete access to all data.
- **Distributed Monitoring of Hosts and Processes:** By starting an agent program on each host to be monitored, Monitor gains the ability to monitor machine statistics.
- **License Management:** Remotely perform start/stop/reread of license servers, as well as manage licensing files, all from within the web interface.

Editions

Monitor is available in two editions, each of which is enabled by a different license:

- **Monitor:** Complete monitoring and reporting capabilities.
- **Monitor Basic:** Monitoring capability only, typically used in conjunction with Accelerator to provide license availability data to its job scheduler. This edition encrypts historical data so that it can be decrypted and used in case a full license is acquired at some point in the future.

Licensing

Monitor is licensed per the number of concurrent unique users that are detected across all of the monitoring that is being performed. As with all Altair Accelerator products, a server license is also required.

Theory of Operation

For monitoring live license servers, data collection jobs, such as `lmstat`, are executed periodically (every 30 seconds as a default, but configurable). When a checkout is detected in the data collection job output, it is stored in the Monitor server memory. The checkout remains active as long as it is detected in subsequent samples that are taken. Once the checkout disappears from the job output, it is considered checked-in by the server and is written to a [checkout data file](#) for the current day. As new data is written to the data file, it is loaded into the database to use for historical reporting.

For debug log parsing (supported for FlexNet Publisher only), parsing jobs are executed periodically (every 12 hours as a default) and denial data is extracted and written to a [denial data file](#) for the current day, organized in a tag/daemon/server directory structure. Optionally, checkout data can also be obtained from the debug log, which will be stored in the checkouts data directory (`licmon.swd/`

`data/checkouts`), also organized in a `tag/daemon/server` directory structure. In both cases, the server portion of the structure will be "master" by default, unless a specific host was configured for the debug log parsing job.

There are two different methods for monitoring unlicensed tools. The first method utilizes a wrapper script or binary to interact with Monitor to perform license counting and restriction, while the second method utilizes a monitoring agent running on every machine that may execute the unlicensed tool. The agent sends information about the processes running on the machine, which is used to detect running tools that have been configured to be monitored.

Interfaces

The primary interface for Monitor is the web interface. The home page, located at the default URL of `http://<HOSTNAME>:5555/cgi/ftlm.cgi` shows an overview of current license activity. The default port requires the user to login. If login is not wanted or needed, a read-only port is also provided that allows for anonymous access. The default read-only port is 5556. The interface utilizes a tab-based interface, which can be used to navigate to a number of views and reports of both current and historical usage, as well as to perform administration of the Monitor implementation.

Monitor also includes a command line interface (CLI), as well as a Tcl-based application programming interface (API). The CLI can be used to obtain license availability and utilization data, as well as create and execute batch-mode reports. The API can be used in any custom application where license availability and utilization data may be needed.

Monitor System Diagram

The following diagram illustrates the Monitor system and data paths.

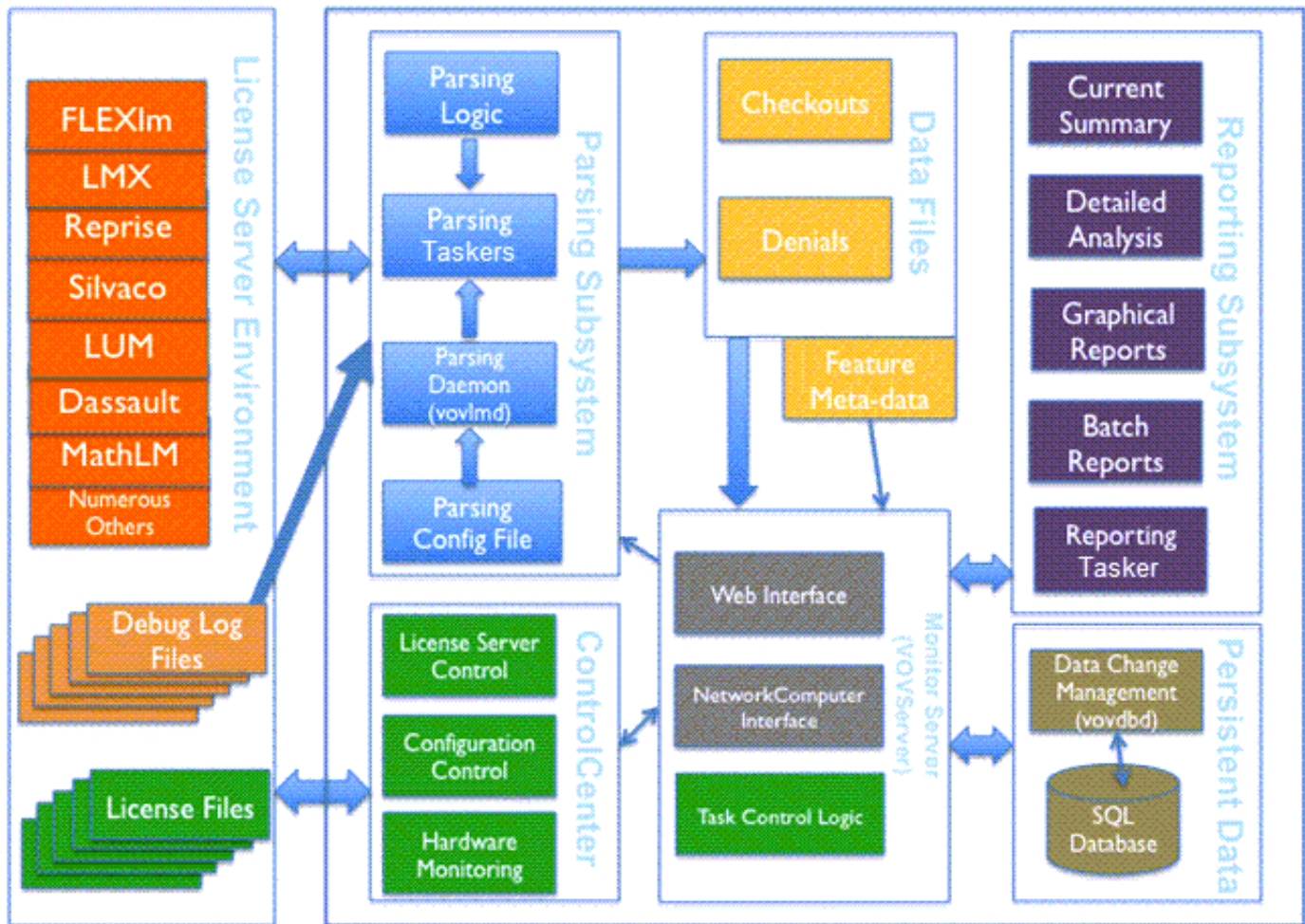


Figure 1:

Monitor Components

All Altair Accelerator products are based on a client/server architecture, provided by the "VOV" framework. The server component of this architecture is called `vovserver`, which provides the Monitor server functionality (web server, job scheduler, etc.) by running a VOV project named `licmon`. The server makes use of a server working directory (SWD) to store its configuration, state and output files. For Monitor, the default location of the SWD is `$VOVDIR/../../licmon/licmon.swd`. The server component is licensed. In addition to the server, the following components are employed:

- **vovlmd:** A daemon that creates data collection jobs that are managed by the job scheduler based on the configuration entered by the administrator.
- **vovtasker:** Processes that execute jobs for the Monitor job scheduler. To minimize latency issues, the workload is distributed amongst multiple vovtaskers. Monitor launches the following vovtaskers on the primary host, by default:
 - `batchreporter`: responsible for completing any background reporting tasks.
 - `logparser`: reads debug logs from license daemons.
 - `maintainer`: completes various tasks as assigned by the vovserver.

- **parser:** runs status on defined license daemons and collects feature, expiration, capacity, checkout, and denial information.

If needed, additional parser taskers can be added on a remote host to increase the scalability of the implementation.

- **vovnotifyd:** A daemon that runs system health checks and e-mails the administrator, and in certain cases, users who trigger events, to provide notification of events.
- **vovdbd:** A daemon that is responsible for monitoring Altair Accelerator product data files for changes and loading them into the database.
- **periodic jobs:** Scheduled jobs that are responsible for automating tasks. These jobs can be configured to run at a specific interval and/or a certain day/time, similar to UNIX cron. Periodic jobs can be configured via the **Configuration Information** page. For a list of periodic jobs that are part of Monitor, see [Maintenance Tasks](#).
- **agents:** Monitor agents (Monitor-Agents) add functionality to Monitor in a number of areas, including:
 - **Control:** License server management facility with full daemon control, license file management, editing, and version tracking.
 - **Network Monitoring:** Monitoring of license server hosts and host processes.

Enable CLI Access on UNIX

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

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

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

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

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

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

- vovrc.csh
- vovrc.sh

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

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

Verify Access to Altair Accelerator Products

After making the changes to source the helper files, and then logging in, you can check that needed environment variables are set properly by looking at environment variables to note that the PATH value contains a reference to the folders in the release which hold programs, and to note that VOVDIR and VOVARCH are set. VOVDIR should contain the path to the installation. VOVARCH should contain the name that matches the machine type it is running on.

1. At the command prompt, enter the following:

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

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

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

```
% vovarch
```

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

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

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

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

or

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

Enable Altair Accelerator for Non-Interactive Shells

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

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

Early exit from non-interactive shell login CSH script:

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

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

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

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

Source helper file before exit in login CSH script:

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

Enable the Shell to Communicate With a Running Product Server

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

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

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

Troubleshooting the UNIX Setup

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

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

```
% csh -c vovarch
```

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

Enable CLI Access on Windows

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

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

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



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

Method 1: Use Windows Explorer to Set Command Line Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Verify Context Is Working

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

Beyond just looking, you can try running a program as a positive test that the context is set up properly. One program that is simple to run and is available with all Altair Accelerator products is

vovarch. This program reports on the machine type on which it is running. A side effect is that it tests that needed environment variables are set properly. If the environment is valid, the program will run and report the correct machine type. If the environment is not valid, the program will not work.

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

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



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

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

Enable the Command Prompt to Communicate With a Running Product Server

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

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

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

Use Monitor Help

Monitor documentation is available in HTML and PDF format.

Access the Help when Monitor is Running

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

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

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


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

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

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

Access the Help when Monitor is not Running

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

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

Access the Help PDF Files

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

Access the Help via the Command Line

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

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


Access the Help via the vovshow Command

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

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

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

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

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

Server Operation

Start Altair Monitor Manually on UNIX

1. Ensure that the license key file is in place (or for customers using a RLM-based licensing environment, ensure that the RLM server is running and Altair licenses are being served).
2. If starting from a UNIX shell, setup the environment to include the Altair Accelerator commands:
 - UNIX: source the shell-specific `vovrc` script found in the Altair Accelerator product installation, in the `common/etc` directory.

After the environment has been setup, start Monitor using the `lmmgr start` command. The start command accepts options to override the port, SWD location, and project name, if desired.

The following shows the usage syntax of the `lmmgr` utility.

Immgr

Altair Monitor management script. Execute on the server. Some commands require enablement by first typing "vovproject enable licmon". If a project name other than licmon is used, use that name instead.

```
lmmgr: Usage Message

DESCRIPTION:
  Altair Monitor management script. Execute on the server. Some commands
  require enablement by first typing "vovproject enable licmon".
  If a project name other than licmon is used, use that name instead.

USAGE:
  % lmmgr <COMMAND> [OPTIONS]

COMMANDS:
  deobfuscate      -- Deobfuscated checkout data that was obfuscated
                   due to license violation. Requires enablement and
                   a valid license file with a proper user count for
                   the dates that contain obfuscated data.
  loaddb           -- Fully load the database. Requires enablement.
  loadremotedata   -- Load data that was copied in from remote instances
                   into the database. Requires enablement.
  reset           -- Resets LM. Requires enablement.
  start           -- Starts LM. The project name is the -name option
                   value if present. Otherwise, licmon.
  stop            -- Stops LM. The project name is the -name option
                   value if present. Otherwise, licmon.

START OPTIONS:
  -block          -- Do not return to the shell or command prompt
                   after starting. This is only useful, and required,
                   when starting Monitor as a Windows service.
  -dbhost <host>  -- Host for database.
```

```

-dbport <port>      -- Port of the database to listen for connections.
-dbroot <path>      -- Path, local to database host, for database files.
                    -- If either -dbhost or -dbroot is left unspecified,
                    -- the database and historical reporting will be
                    -- unavailable until configured in the admin UI.

-dir <dir>          -- Choose top directory (it must exist).
                    -- Default $VOVDIR/../../licmon

-h, -help          -- Print this usage message.
-name <name>        -- Name of the Monitor server project name. Used
                    -- by the start and stop commands.

-port <port[:port]> -- Colon separated port list wherein each element
                    -- may be of the form:
                    N      - An integer specifying the exact port
                           -- number to use.
                    automatic - Calculates hash of project name to
                           -- derive a single port number.
                    any      - Calculates hash of project name to
                           -- derive starting port number. Ports
                           -- are checked sequentially up to
                           -- 30000.
                    Projects will not start if all specified ports
                    are unavailable (default 5557).

-product <product>  -- Product type, one of: lm (default) or lms.
-removelockfile      -- Delete server lock file if it already exists.
-roport <port>       -- Web interface guest access port (default 5556), or
                    -- 0 if no anonymous guest access port should be
                    -- enabled.

-v                  -- Increase verbosity.
-webprovider <prov> -- Web provider: internal or nginx. Default: internal.
-webport <port>      -- Colon separated port list wherein each element
                    -- may be of the form:
                    N      - An integer specifying the exact port
                           -- number to use.
                    automatic - Calculates hash of project name to
                           -- derive a single port number.
                    any      - Calculates hash of project name to
                           -- derive starting port number. Ports
                           -- are checked sequentially up to
                           -- 30000.
                    (Default 5555).

-inport <port[:port]> -- ignored
-eventport <port[:port]> -- Colon separated update port list wherein each
element
                    -- may be of the form:
                    N      - An integer specifying the exact port
                           -- number to use.
                    automatic - Calculates hash of project name to
                           -- derive a single port number.
                    any      - Calculates hash of project name to
                           -- derive starting port number. Ports
                           -- are checked sequentially up to
                           -- 30000.
                    Projects will not start if all specified ports
                    are unavailable (default 5559).

-pubkey <key>       -- Specify the public API key to be added to a new project.
                    -- If -pubkey is not specified, the public key listed in
                    -- ~/.vov/userkey or %PROFILE%/.vov/userkey is added.

STOP OPTIONS:
-f, -force          -- Suppress shutdown confirmation prompt.
-name <name>        -- Name for the server (default licmon).

```

```
LOADDB OPTIONS:
  -start                -- Start time for loading data, (default 1y).

EXAMPLES:
  % lmmgr                -- For this usage message
  % lmmgr start
  % lmmgr start -dir ~/MYFTLM -port 7777
  % lmmgr loaddb
  % lmmgr stop
  % lmmgr stop -force -name licmon833
  % lmmgr deobfuscate
  % lmmgr reset
```

Start Altair Monitor Manually on Windows

1. Download the `lm-win64.exe` program from the release area of the Altair website and save it into a permanent directory, such as `c:\rtda`.
2. Run `lm-win64.exe`.
3. Fill out the information to match the desired installation configuration.

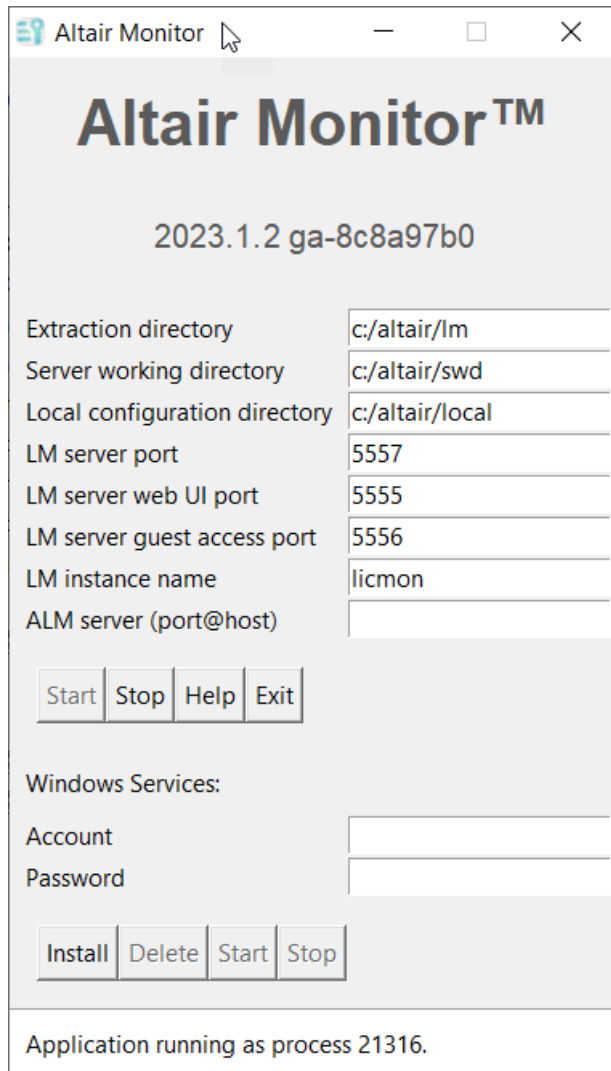


Figure 2:

Monitor can now be controlled manually via the **Start** and **Stop** buttons.


First Startup

The following actions are performed upon first startup:

- The server working directory (licmon.swd) is created at the default location of `$VOVDIR/../../licmon/licmon.swd`, including default configuration files, as well as maintenance tasks for database loading, etc.
- The server process (vovserver) is started.

When the startup process is complete, the web interface will be accessible at the default URL of `http://<HOSTNAME>:5555/cgi/ftlm.cgi`.


Start Monitor Automatically at System Startup - UNIX

 **Note:** If upgrading from a version less than 2015.03, first stop and uninstall the old service before installing the new service. See the *Installation Guide* for the specific version for instructions on doing this.

UNIX

To enable automatically starting `vovtsd` upon UNIX system startup, use the available example `.bat` files as a guide to create your own script, and place it in the appropriate directory. Example startup files are provided in `$VOVDIR/etc/boot`. Choose the one that best fits your scenario.

Start Monitor Automatically at System Startup - Windows

 **Note:** If upgrading from a version less than 2015.03, first stop and uninstall the old service before installing the new service. See the *Installation Guide* for the specific version for instructions on doing this.

1. Download the `lm-win64.exe` program from the Altair website and save it into a permanent directory, such as `c:\rtda`.
2. Run `lm-win64.exe` as administrator by right-clicking it and choosing the **Run as administrator** option.
3. Fill out the information to match the desired installation configuration:

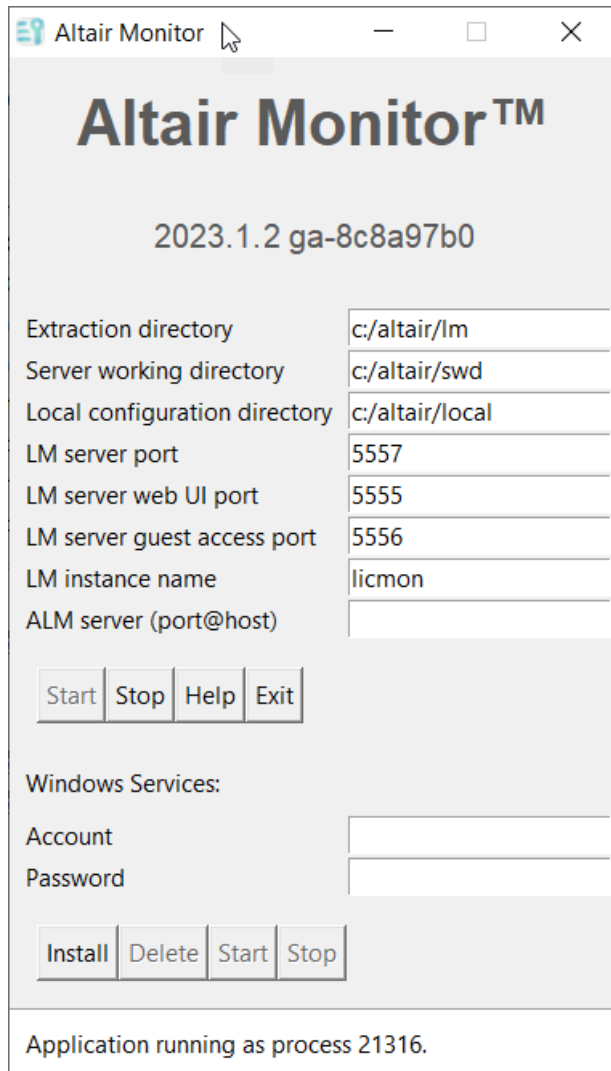


Figure 3:

4. If not using an legacy license key file, make sure to specify the location of the ALM license server or file.
5. Use the Windows Services area to setup and manage a service for controlling Monitor.

Troubleshooting: Images Do Not Display

Occasionally, images may not display correct. The images are retrieved from the readonly port (5556 by default) of the licmon server via HTTP. If your images are not displaying, try the following steps:

1. Check that readonly port is enabled.
2. Check the DNS/NIS setup and `/etc/nsswitch.conf` to be sure that the licmon hostname resolves.

3. Set the environment variable `VOV_HOST_NAME` in `licmon.swd/setup.tcl` to a value that resolves to the Monitor server machine.
4. If an IP address is the only way to access the machine from other hosts, add `VOV_HOST_HTTP_NAME` with the desired IP address to the `setup.tcl` file and perform a reread.
5. To make the change effective immediately, also enter the following commands at the shell. This sets the environment variable in the running `vovserver`.

```
% vovproject enable licmon  
% vovsh -x "vtk_server_setenv VOV_HOST_HTTP_NAME <VALUE>"
```

If you are viewing Monitor over a port-forwarded tunnel through `ssh`, for example, `-L 5555:jaguar:5555`, the host names differ on each end of the connection. The only way to deal with this is to make the Monitor host an alternate name for 'localhost'. For the above example, where the remote host is 'jaguar', your line in the hosts file would be similar to:

```
127.0.0.1 localhost jaguar
```

Reset Altair Monitor

Resetting Monitor consists of clearing all monitoring data, removing the data collection jobs, and recreating them from the configuration files. This may be required if disk-full or network outage situations occur while Monitor is running.

1. To reset Monitor via the web interface, navigate to the **System** page under the **Admin** tab and click on the **reset** button.
2. To reset using the CLI, use:

```
% lmmgr reset
```

Stop Monitor

1. To stop Monitor, use:


```
% lmmgr stop
```

2. To stop without prompting for confirmation, use the option `-force`.

Failover Server Candidates

If a server crashes suddenly, VOV has the capability to start a replacement server on a pre-selected host. This capability requires that the pre-selected host is configured as a failover server.

The configuration instructions follow.

 **Note:** The `vovserverdir` command only works from a VOV-enabled shell when the project server is running.

1. Edit or create the file `servercandidates.tcl` in the server configuration directory. Use the `vovserverdir` command with the `-p` option to find the pathname to this file.

```
% vovserverdir -p servercandidates.tcl  
/home/john/vov/myProject.swd/servercandidates.tcl
```

The `servercandidates.tcl` file should set the Tcl variable `ServerCandidates` to a list of possible failover hosts. This list may include the original host on which the server was started.

```
set ServerCandidates {  
    host1  
    host2  
    host3  
}
```

2. Install the `autostart/failover.sh` script as follows:


```
% cd `vovserverdir -p .`  
% mkdir autostart  
% cp $VOVDIR/etc/autostart/failover.sh autostart/failover.sh  
% chmod a+x autostart/failover.sh
```

3. Activate the failover facility by running `vovautostart`.

```
% vovautostart
```

For example:

```
% vovtaskermgr show -taskergroups  
ID          taskername      hostname      taskergroup  
000404374   localhost-2      titanous      g1  
000404375   localhost-1      titanous      g1  
000404376   localhost-5      titanous      g1  
000404377   localhost-3      titanous      g1  
000404378   localhost-4      titanous      g1  
000404391   failover         titanous      failover
```

 **Note:** Each machine listed as a server candidate must be a vovtasker machine; the vovtasker running on that machine acts as its agent in selecting a new server host. Taskers can be configured as dedicated failover candidates that are not allowed to run jobs by using the `-failover` option in the taskers definition.

Preventing jobs from running on the candidate machine eliminates the risks of machine stability being affected by demanding jobs. The `-failover` option also enables some failover configuration validation checks. Failover taskers are started before the regular queue taskers, which helps ensure a failover tasker is available as soon as possible for future failover events.

The `-failover` option keeps taskers up and running without the need for jobs to be running (in fact, jobs are not allowed to run on them). This allows them to participate in the server election process and start up vovserver without introducing a competition for resources.

Without `-failover`, the taskers will be normal ones, which can run jobs, and in fact, they *must* be running jobs at the time of the vovserver kill/crash because without no jobs, the taskers will exit:

```
if ( ss_activeTransitions <= 0  && (! isInFailoverGroup ) ) {  
    addLog( "No jobs running: no need to keep running." );  
    goto tasker_exit;  
}
```

The easiest way to ensure the tasker is in the failover group is to use the `-failover` option.

Refer to the tasker definition documentation for details on the `-failover` option.

How vovserver Failover Works

If the vovserver crashes, after a period of time, the vovtasker process on each machine notices that it has had no contact from the server, and it initiates a server machine election.

In this election, each vovtasker votes for itself (precisely, the host that this particular tasker runs on) as a server candidate. The election is conducted by running the script `vovservsel.tcl`.

After the time interval during which the vovtaskers vote expires, (default 60 seconds) the host that appears earliest on the list will be selected to start a new vovserver.

In the following example, the `servercandidates.tcl`, file contains three hosts:

```
set ServerCandidates {  
    host1  
    host2  
    host3  
}
```

When the server crashes, if there are vovtaskers running on host1, host2 and host3, then these hosts will be voted as server candidates. Then host2 will be the best candidate and a new vovserver will be started on host2. This server will start in crash recovery mode.



Note: For failover recovery to be successful, an active `vovtasker` process must be running on at least one of the hosts named in the `ServerCandidates` list. Usually, these vovtaskers have been defined with the `-failover` option so they can not accept any jobs, and are members of the `failover` taskergroup.

The failover vovserver will read the most-recently-saved PR file from the `.swd/trace.db` directory, and then read in the transactions from the 'cr*' (crash recovery) files to recover as much of the pre-crash state as possible.

The vovserver writes a new `serverinfo.tcl` file in the `.swd` that vovtaskers read to determine the port and host. When it starts, the failover vovserver appends the new host and port information to the `$NC_CONFIG_DIR/<queue-name.tcl>` as well as to the `setup.tcl` in the server configuration directory. The vovserver then runs the scripts in the autostart directory. This should include the `failover.sh` script, which resets the failover directory so that failover can repeat. This script removes the registry entry, and removes the `server_election` directory and creates a new empty one. At the end, it calls `vovproject reread` to force the failover vovserver to create an updated registry entry.

The failover vovserver remains in crash recovery mode for an interval, usually one minute, waiting for any vovtaskers that have running jobs to reconnect:

- For Accelerator, Accelerator Plus, Monitor and Allocator, vovtaskers wait up to 4 days for a new server to start.
- For FlowTracer, vovtaskers wait up to 3 minutes for a new server to start.

After reconnecting to vovserver, vovtaskers automatically exit after all of their running jobs are completed. After the vovserver transitions from crash recovery mode to normal mode, it will try to restart any configured vovtaskers that are not yet running.

Any of the following conditions will prevent successful failover server restart:

- The filesystem holding the `.swd` directory is unavailable.
- The file `servercandidates.tcl` does not exist.
- The `ServerCandidates` list is empty.
- There is no vovtasker running on any host in the `ServerCandidates` list when the server crashes.
- The `autostart/failover.sh` script file is not in place.

In this case, the failover server will not be automatically started; the server will have to be manually started.

Tips for Configuring Failover

Following are tips for failover configuration:

- Make the first failover host the regular one. This way, if the vovserver dumps core or is killed by mistake, it will restart on the regular host.
- Configure special vovtaskers only for failover by passing the `-failover` option to `vtk_tasker_define`.
- Test that failover works before depending on it.

Migrating vovserver to a New Host

The failover mechanism provides the underpinnings of a convenient user CLI command that can be used to migrate vovserver to a new host:

```
ncmgr rehost -host NEWHOST
```

The specified NEWHOST must be one of the hosts eligible for failover of vovserver.

Crash Recovery Mode

Crash Recovery Mode is activated the next time the server is restarted, if the server was not shut down cleanly. Crash Recovery Mode is part of the Failover Server capability of VOV, which is mainly used in Accelerator. This capability allows VOV to start a new server to manage the queue of a server which has crashed unexpectedly.

When you shut down an Monitor instance cleanly using the `lmmgr stop` command, the server will save its database to disk just before exiting. When the server is restarted, it will read the state of the trace from disk, and immediately be ready for new work.

Sometimes the vovserver will be stopped unexpectedly, such as due to a hardware problem like a machine crash or memory exhaustion. In such cases, the server will not have a chance to save the project database before terminating.

- In VOV, the main concern is usually the state of the trace, which stores the status all the jobs in your project.
- In Accelerator, there is no trace, and the important thing to preserve is the state of the queue, so jobs do not lose their position and need to be re-queued in the case of a server crash.

Journal Files

The vovserver keeps crash recovery journal files of the events that affect the state of the server. These 'CR' files are flushed whenever the trace data are saved to disk. During crash recovery, the vovserver first reads the last saved state of the trace from the disk data, then applies the events from the CR files.

Crash Recovery Restart

When the server is next restarted after such a crash, the server enters what is called Crash Recovery Mode, which usually lasts about two minutes, but may take longer if the CR files are very large. During this period:

- The server waits for vovtaskers with running jobs to reconnect.
- No jobs are dispatched.
- The server does not accept VOV or HTML TCP connections from `vovsh` or browser clients.
- At the end, the server performs a global sanity check.
- A `crash_recovery_report <timestamp>` logfile is written. It logs any jobs lost by the crash recovery sequence.

If the server is properly shut down, the next time it restarts, it will not enter Crash Recovery Mode, and will be immediately functional.



Note: If the sanity check command cannot connect, the server is still recovering its state from the DB and CR files. Check the size of the `vnc.swd/trace.db/CR*` files.

Example commands:

```
% vovproject enable <project-name>
% vovproject sanity
```

Daemon Operation

Although the Monitor daemons are started automatically upon startup, there may be times where a daemon should be restarted. This section describes how to control the Monitor daemons.

Web-based Operation

Daemons can be controlled from the web interface by visiting the Daemons administration page:

LicenseMonitor System Daemons

View the status of, and control, the various daemons that LicenseMonitor requires for operation.











Type	Daemon	Config URL	Config File	Info File	Status	Action
daemon	vovresourced		Show config file	Show info file	DOWN	 
daemon	vovnginxd				NOT CONFIGURED	
daemon	voveventstatd				NOT CONFIGURED	
daemon	vovlmd		Show config file	Show info file	DOWN	 
daemon	vovnotifyd	config	Show config file		DOWN	 
daemon	vovdbd		Show config file	Show info file	DOWN	 

Figure 4: Daemons

Control daemons by clicking the desired action icon (start or stop). The configuration files for each daemon can also be viewed from this page.

The vovtriggerd Daemon


This particular daemon should not be running under normal circumstances. When running, Monitor enters a debugging mode in which the output of each sampling job run is archived in the `licmon.swd/logs/parser` directory so that the history can be "played back" by Altair support if needed. Such cases would be when there is an intermittent issue that is difficult to reproduce but happens over time. Monitor can potentially use large amounts of disk space when running in this mode, so make sure that the `vovtriggerd` daemon is running only when needed.

Daemon Manager Utility

Although the Monitor daemons are started automatically upon startup, there may be times where a daemon should be restarted. This section describes how to control the daemons from the command line (CLI).

CLI-based Operation

To control daemons in the CLI, a utility is provided called `vovdaemonmgr`

 **Note:** Daily log files generated by system daemons are now automatically compressed (non-Windows only).

vovdaemonmgr

This is the command to show the status of daemons. You can also use this command to start/stop the daemons.

vovdaemonmgr: Usage Message

This is the command to show the status of daemons. You can also use this command to start/stop the daemons.

NOTE: This command can only be used by the owner of the vovserver and on the machine where the vovserver runs.

USAGE:

vovdaemonmgr <SUBCOMMAND> [OPTIONS] [daemonsList]

SUBCOMMAND is one of:

list	-- List the configured daemons. (list -all for all daemons)
restart	-- Restart the specified daemons.
show	-- Show the status of the specified daemons.
status	-- Same as 'show'.
start	-- Start the specified daemons.
stop	-- Stop the specified daemons.

[daemonsList] is optional. When omitted, act on all daemons.

OPTIONS:

-f	-- Force flag (for start only). If the start subcommand has an explicit daemonsList, the specified daemons will be started even if not configured.
-h	-- Help usage message.
-v	-- Increase verbosity.
-retry <N>	-- For NIS, control how many retries to attempt, default 0
-wait <N>	-- For NIS, control how long to wait between retries, default 0s

EXAMPLES:

```
% vovdaemonmgr list
% vovdaemonmgr list -all
```

```
% vovdaemonmgr status
% vovdaemonmgr status vovpreemptd
% vovdaemonmgr start vovnotifyd
% vovdaemonmgr start -force vovresourced
% vovdaemonmgr start -force -v -v vovresourced
% vovdaemonmgr stop vovlad
```

Database

Monitor stores historical information about jobs in a relational database. The database is fully integrated and managed as part of Monitor. This section provides an overview of the components that run and manage the database.

The database, `vovdb`, is based on PostgreSQL™, a performant and reliable open-source database engine with decades of wide-scale use. There are two ways to configure and manage `vovdb`: through the web UI and via the command line.


It should never be necessary to run `vovdb` directly; once configured and enabled, Monitor will automatically start and stop the database as needed.

Daemon

Certain database tasks are managed via the `vovdbd` daemon: a background process that can load data, perform maintenance, or trigger backups. These tasks can be configured through the database administration web UI.

Refer to the *Daemons* documentation for more information on how Monitor manages daemons.

Tasker

 **Note:** The information in this section does not apply to Windows-based installations.

A `vovdbd` tasker will be created as necessary to manage the database. This occurs when certain database commands (starting, stopping, and performing backups) are run from a host different than the configured database host.

This typically occurs when the database is configured with a remote host, but may also occur with a local database if [command line](#) operations are run from another machine. In either case, the command is automatically spawned to the `vovdbd` tasker as an interactive job; the command behaves identically whether run locally or remotely.

The `vovdbd` tasker does not consume a license, and is reserved for database-related jobs only.

Set Up

For the first start of Monitor, click **Admin > System > Database** to configure the database. You must have ADMIN privileges on the Monitor server to configure and control the database.

Database Configuration

Use the page to configure and control the database, check database status, and review database statistics.

Database Location

Port: 11505

Port of the database to listen for connections.

By default, the system initialized a random port to use.

Path: C:/altair/db2

Path to the database storage location as seen by the specified host.

It is highly recommended to use a path that is located on the specified host's local disk for performance and reliability reasons.

It is also recommended to include this product's instance name (licmon) in the path to help identify the directory's purpose.

Edit Location

Database Control

ON

Database Tasks

These tasks require the [vovdbd daemon](#).

Daemon status: down

☒ Enable Data Loader

For LicenseMonitor this controls loading of sampled checkouts only.

Checkouts and denials gathered through debug log parsing are loaded by the parsing job regardless of this setting.

Reset

Save

Figure 5: System Database Configuration

Database Location

In this section, specify the host (if a separate database host is desired), port (if a specific port is desired), and the path to the database directory as seen by the specified database host. The configuration page will default to using the local host, a port randomly selected, and a path that is located in the installation area.

If configuring a separate database host, it is assumed that an SSH connection can be made between the Monitor server host and the remote database host without a password prompt appearing. Consult your IT organization to set up an SSH key environment if needed.



Note:

- Separate database hosts are not supported for Windows-based installations.
- It is highly recommended to utilize the local file system for the database directory. While remote file systems will work, the resulting performance will be far less than with the local file system.

The database host and path can also be configured from the command line. See *Configure the Database from the Command Line* for more information.

Database Control

After the database host and location have been specified, drag the slider switch to the ON position to create and start the database. The database will now be started automatically upon each start of Monitor.

To stop the database, drag the slider switch to the OFF position. When the database is not running, data will continue to be collected by Monitor but it will not be loaded until the database is restarted.

The database can also be started and stopped from the command line. See *Configure the Database from the Command Line* for more information.

Database Tasks


This section allows configuration of the database tasks. These tasks cannot be configured from the command line.

The tasks are run by the `vovdbd` daemon. Status is shown and updated every 30 seconds. For more information, refer to *Altair Accelerator Daemons*.

The Data Loader checkbox controls the automatic loading of jobs. The loader runs continuously, but is automatically disabled when doing maintenance or backup.

The Maintenance settings enable daily database maintenance and specify the time window to perform maintenance. The database is still available during the maintenance window; however performance may be impacted. For this reason, it is best to schedule maintenance for off-peak hours.

Database Backups enables automated backups. The frequency of the backups, backup location, time window in which to start the backup, and how many backups to keep can be specified. As with maintenance, the database is still available while a backup is being made, however performance will be impacted.

 **Note:** Network storage is acceptable for database backups.

Configure the Database from the Command Line

The command line utility `vovdb_util` can be used to configure most aspects of the Monitor database. You must have ADMIN privileges on the Monitor server to configure and control the database.

To execute the utility, first set up the CLI:

```
% vovproject enable vnclicmon
```

vovdb_util

Utilities for use with the VOV database.

```
vovdb_util: Usage Message  
DESCRIPTION:
```

Utilities for use with the VOV database.

USAGE:

```
% vovdb_util <COMMAND> [COMMON OPTIONS] [OPTIONS]
```

COMMANDS:

backup <dest>	-- Backs up the VOV database to the specified destination. This command must be run on the same machine as the database. If the destination exists it must be empty. The database must also be running.
clearcfg [-noconfirm]	-- Resets the VOV database configuration to the initial, unconfigured state. Pass -noconfirm to skip confirmation.
exportconfig <fileout>	-- Exports VOV database configuration properties DB_* to obfuscated file.
exportpasswords <fileout>	-- Exports VOV database passwords from project to obfuscated file.
help	-- Shows help information.
importconfig <filein>	-- Imports VOV database configuration properties DB_* from obfuscated file.
importpasswords <filein>	-- Imports VOV database passwords from obfuscated file to project.
configure [-v] [-reset] [-noconfirm] [-connection:secure=0/1] <host> <root> [<port>]	-- Sets the host, root data path, port and SSL for the VOV database. Pass -reset to overwrite existing settings. Pass -noconfirm to skip confirmation. Connection argument sets SSL mode of communication with database. Default set by server is 0 i.e. SSL is disabled.
showcfg	-- Prints out current VOV database configuration.
startdb	-- Starts the VOV database. Will restart a running database.
status	-- Prints current VOV database status.
stopdb	-- Stops the VOV database.
upgrade [-noconfirm] [-sdb sourcedir] [-spgsw pg_software_dir]	-- Upgrades the VOV database to use the newest version of the PostgreSQL engine. Options: -noconfirm to skip confirmation -sdb to specify path to Source database -spgsw path to older version of PG binaries compatible with Source database

The following commands are only supported for Accelerator and Monitor.

dump [-pre201509] [-start <YYYYMMDD>] [-end <YYYYMMDD>]	-- Generate data files, optionally limited to the start and end times specified. Pass -pre201509 to dump the database that was used prior to 2015.09 (or beyond.)
trim <YYYYMMDD>	-- Deletes data prior to the given date.

COMMON OPTIONS:

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

Database Configuration Options

Database configuration is handled by the `vovdb_util configure` command. Pass `-reset` to overwrite an existing configuration. Pass `-noconfirm` to skip confirmation.

To show the current configuration, use `vovdb_util showcfg`. To clear an existing configuration, use `vovdb_util clearcfg`.

Some examples:

```
% vovdb_util configure localhost /data/rtda/licmon
vovdb_util 08/17/2024 16:15:03: message: Setting VOV database configuration to:
Host:      srv1
Data Path: /data/rtda/licmon
Set configuration (yes/no)? yes
vovdb_util 08/17/2024 16:15:08: message: Configuration saved.

% vovdb_util configure -reset -noconfirm localhost /data/rtda/db/licmon
vovdb_util 08/17/2024 16:15:53: message: Configuration saved.

% vovdb_util showcfg
vovdb_util 08/17/2024 16:16:00: message: The VOV database configuration is:
Host:      srv1
Data Path: /data/rtda/db/licmon
Status:    stopped

% vovdb_util clearcfg -noconfirm
vovdb_util 08/17/2024 16:15:24: message: Configuration has been cleared.
```

Database Control Options

Once the database is configured, it can be controlled with the `vovdb_util startdb` and `stopdb` commands. There will be additional output the first time the database is started as the on-disk structure is created.

Starting the database:

```
% vovdb_util startdb
vovdb 08/17/2023 16:22:55: message: Creating database
The files belonging to this database system will be owned by user "admin".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /data/rtda/db/licmon/dbdata9_4 ... ok
creating subdirectories ... ok
<snip>
vovdb 08/17/2023 16:22:58: message: Starting database
vovdb 08/17/2023 16:22:58: message: LOG:  database system was shut down at 2023-08-17
16:22:56 CDT
```

```
vovdb 08/17/2023 16:22:58: message: LOG: MultiXact member wraparound protections are now enabled
vovdb 08/17/2023 16:22:58: message: LOG: database system is ready to accept connections
vovdb 08/17/2023 16:22:58: message: Database engine is ready.
vovdb 08/17/2023 16:22:58: message: Configuring database...
vovdb 08/17/2023 16:22:58: message: Creating user 'rtdamgr'
vovdb 08/17/2023 16:22:58: message: Creating user 'rtdausr'
vovdb 08/17/2023 16:22:58: message: Creating database 'rtda'
vovdb 08/17/2023 16:22:58: message: Database configured.
vovdb 08/17/2023 16:22:58: message: Loading schema for LicenseMonitor...
vovdb 08/17/2023 16:22:58: message: Creating table 'metadata'...
vovdb 08/17/2023 16:22:58: message: Granting RO privileges to rtdausr...
vovdb 08/17/2023 16:22:58: message: Creating table 'loadinfo'...
vovdb 08/17/2023 16:22:58: message: Granting RO privileges to rtdausr...
<snip>
vovdb 08/17/2023 16:22:59: message: Schema loaded
vovdb 08/17/2023 16:22:59: message: Database is ready.
vovdb_util 08/17/2023 16:22:59: message: Database started.
```

Stopping the database:

```
% vovdb_util stopdb
vovdb_util 08/17/2023 16:31:22: message: Stopping database...
vovdb_util 08/17/2023 16:31:23: message: Database stopped.
```

Database Tasks

At present, database tasks such as the automatic loader, daily maintenance, and automatic backups can only be configured through the web interface. Refer to *Set Up* for more information.

Database Engine Versions and Upgrades

Monitor includes version 14.4 of the PostgreSQL database engine that is used by the VOV database daemon `vovdb`. This section describes how to determine if a database upgrade will be needed when upgrading the Accelerator product software.

Database Engine Versions

PostgreSQL 14.4 is an improved and updated database version. An existing database created with PostgreSQL 9.4 or 9.6 cannot be used directly in 14.4; it must be upgraded. The upgrade process is described in [Database Upgrades](#). When you plan to upgrade an existing Monitor project from a Monitor software version with PostgreSQL 9.4 or 9.5 to a Monitor version with PostgreSQL 14.4, you will need to choose a database upgrade strategy from the options described in the *Altair Accelerator Software Installation Guide*.

To determine the running Monitor project's PostgreSQL version, visit the **System > Database Configuration** web UI page. If the PostgreSQL version is 14.4, then you may simply upgrade Monitor via the Software Upgrade Instructions section in the manual.

Load Data

This section details the methods that Monitor uses to load data into the database, and the configuration options available to control which data is loaded.

Automatic Loading

Checkout data files generated from sampled data will be automatically and continuously loaded if the daemon-based loader is enabled.

Checkouts and denials generated from debug log parsing are loaded by the parser itself. For more information on debug log parsing, refer to [Debug Log Monitoring](#).

Manual Loading

Manual loading of job data files can be accomplished by using the `vovsql_load_checkouts` and `vovsql_load_denials` command line utility.

To execute the utility, set up the CLI:

```
% vovproject enable licmon
```

The checkout loader's syntax is:

```
vovsql_load_checkouts: Usage Message

USAGE:
    % vovsql_load_checkouts [OPTIONS] <listOfFiles>

OPTIONS:
    -h                -- This help.
    -v                -- Increase verbosity.
    -q                -- Quiet.
    -origin <N>       -- Specify origin of data (see below)
    -f                -- Same as -force.
    -force            -- Force reloading of data files.
    -convert          -- Convert the old data files to current format.

REQUIRED ARGUMENTS:
    <listOfFiles>     -- A list of checkout data files. Can also be specified
                        as a glob expression (e.g. checkouts/2009.12.*)
                        or can contain the tokens @TODAY@ and @YESTERDAY@.

ORIGIN:
    1                -- The data comes from the sampling (default).
    2                -- The data comes from debug log parsing, to be merged.
    6                -- The data comes from debug log parsing,
                        not to be merged.

EXAMPLES:
    % vovsql_load_checkouts          licmon.swd/data/checkouts/2007*
    % vovsql_load_checkouts -force   licmon.swd/data/checkouts/2007*
    % vovsql_load_checkouts -origin 2
      licmon.swd/data/checkouts/MGC/mgcld/master/2010*
    % vovsql_load_checkouts          licmon.swd/data/checkouts/@TODAY@
vovsql_load_checkouts: Usage Message

USAGE:
    % vovsql_load_checkouts [OPTIONS] <listOfFiles>
```

```
OPTIONS:
  -h                -- This help.
  -v                -- Increase verbosity.
  -q                -- Quiet.
  -origin <N>      -- Specify origin of data (see below)
  -f                -- Same as -force.
  -force            -- Force reloading of data files.
  -convert          -- Convert the old data files to current format.

REQUIRED ARGUMENTS:
  <listOfFiles>    -- A list of checkout data files. Can also be specified
                    as a glob expression (e.g. checkouts/2009.12.*)
                    or can contain the tokens @TODAY@ and @YESTERDAY@.

ORIGIN:
  1                -- The data comes from the sampling (default).
  2                -- The data comes from debug log parsing, to be merged.
  6                -- The data comes from debug log parsing,
                    not to be merged.

EXAMPLES:
  % vovsql_load_checkouts          licmon.swd/data/checkouts/2007*
  % vovsql_load_checkouts -force    licmon.swd/data/checkouts/2007*
  % vovsql_load_checkouts -origin 2
    licmon.swd/data/checkouts/MGC/mgcld/master/2010*
  % vovsql_load_checkouts          licmon.swd/data/checkouts/@TODAY@
```

The denial loader's syntax is:

```
vovsql_load_denials: Usage Message

USAGE:
  vovsql_load_denials [OPTIONS] <datafile> ...

OPTIONS:
  -h                -- This help.
  -v                -- Increase verbosity.
  -q                -- Quiet.
  -f, -force        -- Force reloading of data files.
  <logfile>         -- A denial data file.
                    Can be specified as a glob expression.
                    Can be repeated.

EXAMPLES:
  % cd `vovserverdir -p data`
  % vovsql_load_denials ./denials/*/*/2008*
```

Control Which License Usage Data is Loaded

By default, Monitor will load all checkout and denial data into the database. However, sometimes there exists a license feature that is used heavily but is either free or unlimited, and thus generates heavy load on the database while being of little value when generating reports. In this scenario, the feature can be blocked from being loaded into the database to help reduce load on the database.

To block loading of a feature's checkout and denial data, edit the `licmon.swd/vovdbd/config.tcl` file and define `VOVDBD(lm,excludeFromDb)`. The value should be of the format:

```
set VOVDBD(lm,excludeFromDb) { { tag feature } { tag feature } ... }
```

Standard shell wildcards such as `*` and `?` are supported in either the tag or feature name. Some examples:

```
# To exclude "FeatureA" from all tags:
set VOVDBD(lm,excludeFromDb) { { * FeatureA } }

# To exclude "FeatureA" from all tags and "FeatureB" from "Tag2":
set VOVDBD(lm,excludeFromDb) { { * FeatureA } { Tag2 FeatureB } }
```

If at some point it is desired to load previously excluded data, edit `config.tcl` to remove the exclusion, then manually load the data files using the `-force` option to either `vovsql_load_checkouts` or `vovsql_load_denials`.

Export Data

One function of the utility `vovdb_util` is to export the database into jobs data files. The exported files are saved in the `vnc.swd/data/dump` directory.

To execute the utility, first set up the CLI:

```
% vovproject enable licmon
```

vovdb_util

```
vovdb_util: Usage Message
DESCRIPTION:
  Utilities for use with the VOV database.

USAGE:
  % vovdb_util <COMMAND> [COMMON OPTIONS] [OPTIONS]

COMMANDS:

  backup <dest>                                -- Backs up the VOV database to the specified
                                                destination. This command must be run on
                                                the same machine as the database. If the
                                                destination exists it must be empty.
                                                The database must also be running.

  clearcfg [-noconfirm]                        -- Resets the VOV database configuration to
                                                the initial, unconfigured state. Pass
                                                -noconfirm to skip confirmation.

  exportconfig <fileout>                       -- Exports VOV database configuration
                                                properties DB_* to obfuscated file.

  exportpasswords <fileout>                   -- Exports VOV database passwords
                                                from project to obfuscated file.

  help                                         -- Shows help information.

  importconfig <filein>                       -- Imports VOV database configuration
                                                properties DB_* from obfuscated file.
```



```
importpasswords <filein>          -- Imports VOV database passwords
                                   from obfuscated file to project.
configure [-v] [-reset] [-noconfirm] [-connection:secure=0/1] <host> <root>
[<port>]                           -- Sets the host, root data path, port and SSL for
                                   the VOV database. Pass -reset to overwrite
                                   existing settings. Pass -noconfirm to skip
                                   confirmation.
                                   Connection argument sets SSL mode of
                                   communication with database. Default set by
                                   server is 0 i.e. SSL is disabled.
showcfg                             -- Prints out current VOV database
                                   configuration.
startdb                             -- Starts the VOV database. Will restart a
                                   running database.
status                             -- Prints current VOV database status.
stopdb                             -- Stops the VOV database.
upgrade [-noconfirm] [-sdb sourcedir] [-spgsw pg_software_dir]
                                   -- Upgrades the VOV database to use the newest
                                   version of the PostgreSQL engine.
                                   Options:
                                   -noconfirm to skip confirmation
                                   -sdb to specify path to Source database
                                   -spgsw path to older version of PG
                                   binaries compatible with Source database

The following commands are only supported for Accelerator and Monitor.

dump [-pre201509] [-start <YYYYMMDD>] [-end <YYYYMMDD>]
                                   -- Generate data files, optionally limited to
                                   the start and end times specified. Pass
                                   -pre201509 to dump the database that was
                                   used prior to 2015.09 (or beyond.)
trim <YYYYMMDD>                   -- Deletes data prior to the given date.

COMMON OPTIONS:
-v                               -- Increase verbosity.
```

Database Export

Database export is done using the `vovdb_util dump` command. Pass the optional `-start YYYYMMDD` and `-stop YYYYMMDD` to restrict the dump to after the specified start date and before the specified end date.

Dump Pre-2015.09 Databases

If updating Monitor from a version prior to 2015.09, it is possible to pass the `-pre201509` option to `vovdb_util dump` to generate data files from the old database. This can be used to save old data if the original data files are no longer present.

Checkouts

The `licmon.swd/data/dump/checkouts` directory contains the exported checkouts data organized by the data origination (1 = samples, 2 = debug log). The dumped data files can be loaded into a new database using the [vovsql_load_checkouts](#) utility.

When loading the sampled data, no arguments are needed. When loading the data gathered from the debug logs, the `-origin 2` option needs to be passed to the utility to instruct the loader to place them in this origin in the new database.

Denials

The `licmon.swd/data/dump/denials` directory contains the exported denial data. The directory contains subdirectories for each tag that contains denial data in the database. The dumped data files can be loaded into a new database using the [vovsql_load_denials](#) utility.

Database Backup

There are two ways to do direct backups of the database: automatic and manual. In addition, the original source data files may be saved.

Automatic Backups

Automatic backups are configured through the [administration web UI](#). When enabled, the backups are generated in subdirectories of the configured Backup Location named for the time of backup.

If the backup location is configured as:

```
/data/rtda/db_backup/licmon
```

The backup started on August 21st, 2023 at 1:00AM will be located in:

```
/data/rtda/db_backup/licmon/20230821_010000
```

Manual Backups

Manual backups are generated by running the `vovdb_util` utility. Backups are generated directly in the directory specified; it is recommended to specify the time of backup in the directory name for future reference. Example:

```
% vovdb_util backup /data/backup/licmon/2023Aug22_10AM
```

Source Data Backups

In addition to backing up the database directly, the source data files used to populate the database can be saved. The source files are located in `licmon.swd/data` and can be used to rebuild a database from scratch. No utility is provided to save these files.

Restore a Database from Backups

The procedure to restore from backup is the same whether the backup was generated automatically or manually.

1. Ensure the current database is stopped, either through the [web UI](#) or via the [command line](#).
2. Delete or move the existing database.

For example, if the database is located at `/data/rtda/db/licmon`, it can be moved with:

```
% mv /data/rtda/db/licmon /data/rtda/db/vncllicmon.bad
```

3. Copy or move the backup to the original database location.

```
# Using the example of an automatic backup from above.  
% mv /data/rtda/db_backup/licmon/20160821_010000 /data/rtda/db/vncllicmon  
  
# Using the example of a manual backup from above.  
% mv /data/backup/licmon/2016Aug22_10AM /data/rtda/db/vncllicmon
```

4. Restart the database.

Rebuild a Database from Source Data Files

1. Ensure the current database is stopped, either through the [web UI](#) or via the [command line](#).
2. Delete or move the existing database.
For example, if the database is located at `/data/rtda/db/licmon`, it can be moved with:

```
% mv /data/rtda/db/licmon /data/rtda/db/vncllicmon.bad
```

3. Reload the source data files.

```
% vovproject enable licmon  
  
% lmmgr loaddb
```

Tag Administration

The Tag Administration page, found in the Admin tab, provides an interface to managing tags. A tag is defined as a unique identifier for a license server being monitored. Tags are chosen by the Monitor administrator. The actions that are available in the tag administration page are listed below.

Manage Tags in Server Memory

Tags are managed separately between the server memory and the database. This allows for historical data to be independent of what is currently being monitored. Normally, when a tag is removed from the Monitor configuration, the `vovlmd` daemon will detect the configuration change and restart itself, in turn removing the affected tag from server memory. Removal can also be performed manually by using this section on the tag administration page:

Tag Administration (server)

A tag in LicenseMonitor is a unique identifier for a license server that is being monitored. This page provides an interface to administrative functions for managing tags and tag-related data found in the server and in the database.

The below tags are being tracked in server memory. If a tag or daemon has been removed from the configuration but still appears in this table, it can be deleted using the delete button below. If a tag is deleted that still exists in the configuration, it will reappear.

<input type="checkbox"/>	Tag	Daemon	Server	Features	Checkouts	Last Updated
<input type="checkbox"/>	ALM		6200@aap-licsrv.prog.altair.com	11	138	3d17h

?

Showing 1 out of 1 row | Limit rows to display: 100 | [Filter](#) ☐ ignore case

Delete Selected Tags From Server Memory

Figure 6: Tags - Server

Note that any tag that is removed will return upon the next sampling cycle if the configuration was not updated as well.

Manage Tags in the Database

To remove tags from the database, use this section on the Tag Administration page:

Tag Administration (database)

A tag in LicenseMonitor is a unique identifier for a license server that is being monitored. This page provides an interface to administrative functions for managing tags and tag-related data found in the server and in the database.

The below tags are found in the database. To delete a tag and all of its data from the database, use the delete button below. For a tag to not reappear in the database, it must first be removed from the configuration and server memory.

<input type="checkbox"/>	Tag	Features	Checkouts	Denials
<input type="checkbox"/>	AL001	69	151,519	0
<input type="checkbox"/>	AL002	59	144,706	0
<input type="checkbox"/>	ALM	11	6,503	0
<input type="checkbox"/>	BUILT_IN	1	1	0
<input type="checkbox"/>	CDN_TROY	2	72	0
<input type="checkbox"/>	RTDA_ALM	7	24,549	0
<input type="checkbox"/>	RTDA_RLM	12	40,241	0
<input type="checkbox"/>	RTDA_TROY	10	15,863	0

?

Showing 8 out of 8 rows | Limit rows to display: 100 | [Filter](#) ☐ ignore case

Delete Selected Tags From Database

Figure 7: Tags Administration Database - Upper

This operation will not affect the server memory nor the configuration. Note that any tag that is removed from the database will eventually return if the same tag still exists in the server memory and/or the configuration.

Rename Tags

If the initial tag name chosen becomes invalid over time, it can be renamed. When renaming a tag, make sure that the Monitor configuration is also updated to reflect the new name before renaming so that data does not continue to be written into the database under the old tag name. Use this section to rename a tag:

Rename Tag

Rename a tag in the database. If renaming to an existing tag, data will be updated to reflect that tag. If renaming a tag that is actively being monitored, remember to rename it in the configuration as well.

From:

AL001

To:

AL001

- OR -

Rename

Figure 8:

Assign Site

Hierarchically, tags belong to sites. By default, each tag is assigned to the *LocalSite* site. Use this section to change the site for a tag to a new or existing site:

Site Association

Associate a tag with a site. This information is currently presented on this page only.

Tag:

AL001

Site:

LocalSite

- OR -

Assign

Figure 9: Assign Site

Check and Clear Tag Modification History

Tag changes that are performed by using any of the above functions on the tag administration page are tracked and displayed in the modification history section:

Tag Modification History

The below shows a list of operations that have been performed to tags in the database.

Clear Tag History

Figure 10: Tag Modification History

The history can be cleared by clicking **Clear Tag History** button at the bottom of the page.

Manage Data from Different Origins

In FlexNet Publisher monitoring, checkout data can be obtained from two different data sources.

Those two sources are:

- `lmstat` sampling
- debug log

Monitor distinguishes these data sources as origins. The data are stored in the database with indicators for its associated origin. Monitor utilizes the following origins:

Symbolic Name	Description	Corresponding Database Symbolic Values
samples	lmstat-driven samples	1
logs	parsed from debug log	2

Default Origin

The data origin for all historical reports defaults to that of the lmstat "samples" (1). This default can be specified in the `licmon.swd/vovlmd/config.tcl` file, using the following setting:

```
# Fragment of ../vovlmd/config.tcl
set VOVLM(origin) "samples"
# set VOVLM(origin) "logs"
```

License Violation Effects and Actions

Monitor requires a license in order to store and report on historical license utilization information. If at any point a license is not available while the server is running, the server enters a licensing "grace period" of five days and the system will continue to operate as normal.

If the grace period is exceeded, the data will continue to be tracked, but will be obfuscated (encrypted) and will not be loaded into the database. This data will be stored in the checkouts data files just as normal tracking data would, but will appear as an unreadable string for the records that were obtained during the license violation. License violations typically fall within three categories:

- The Altair Accelerator license is expired.
- If using ALM, the license server is down. If using a license key file, the key file is missing. In either of these two cases, all data will be obfuscated until license availability is restored.
- There are more observed users across the various monitors than what the Altair Accelerator license contains. For example, for a license of 50 users, and Monitor detects 56 unique users at a given point in time, the server assigns an index to each unique user and all checkouts for users 51 to 56 will be saved as an obfuscated entry.

Deobfuscation

The server keeps track of all data files that contain obfuscated data. In order to deobfuscate the data, a valid license file must be in place beforehand. If the license violation was due to an expiration or an overage of licensed users, the license file must contain a back-dated start date and/or an increased number of licensed users in order to allow for deobfuscating the historical records during the time of the violation. Monitor provides two methods that can be used to deobfuscate the data in these files and load the resulting output into the database:

Web-based Interface

To check for the existence of obfuscated data and deobfuscate it, visit the **Database Information** area of the **System** page, located in the **Admin** tab. If obfuscated data has been generated by the system, the data files that contain the obfuscated data will show up at the bottom of the page, along with a button to use to perform the deobfuscation:

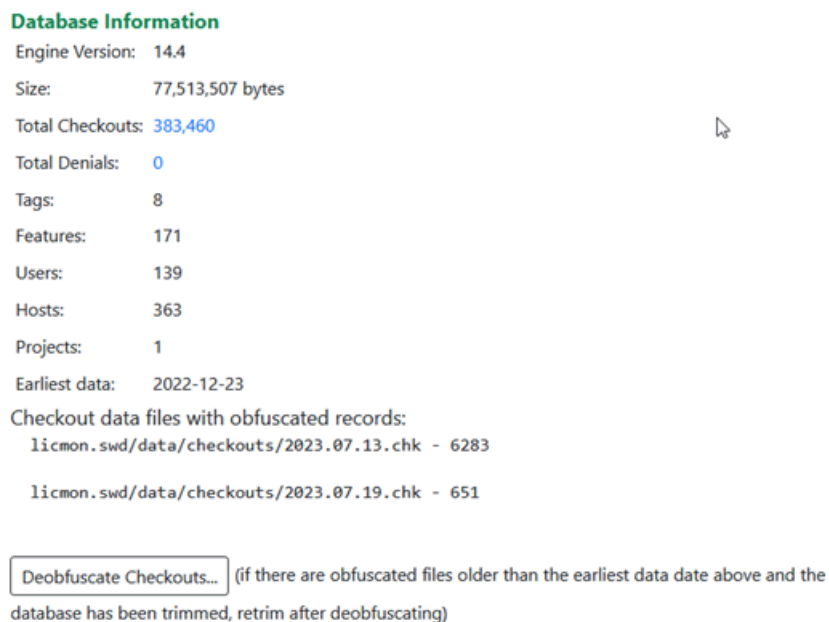


Figure 11: Database Information Page

Command Line Interface

In the CLI, the `lmmgr deobfuscate` command can be used. To use the command, first setup the CLI:

```
% vovproject enable licmon
```

After setting up the CLI, the command can be executed:

```
% lmmgr deobfuscate
```

If more precise control over the deobfuscation process is required, use the `ftlm_deobfuscate` utility instead:

```
ftlm_deobfuscate: Usage Message
```

DESCRIPTION:

Utility to deobfuscate and load checkouts that have been obfuscated by LicenseMonitor during situations where the product license was either unavailable, exceeded, or expired.

If no file is specified, the utility automatically detects and deobfuscates all log files that contain obfuscated data.

USAGE:

```
% ftlm_deobfuscate [OPTIONS]
```

OPTIONS:

-h	-- Show this help.
-v	-- Increase verbosity.
-do	-- Perform deobfuscation and load data into DB.
-k	-- Specify license key file path. Default is to search in the following order: licmon.swd/license.lm.key licmon.swd/license.key VOVDIR/local/license.lm.key
-show	-- Show files that are known to have obfuscated data.

EXAMPLES:

```
% ftlm_deobfuscate -do  
% ftlm_deobfuscate -show  
% ftlm_deobfuscate -do 2010.04.01  
% ftlm_deobfuscate -do 2010.04.*
```


Database Schema

The SQL schema used by this version of Monitor is visualized in the following entity-relationship diagram.

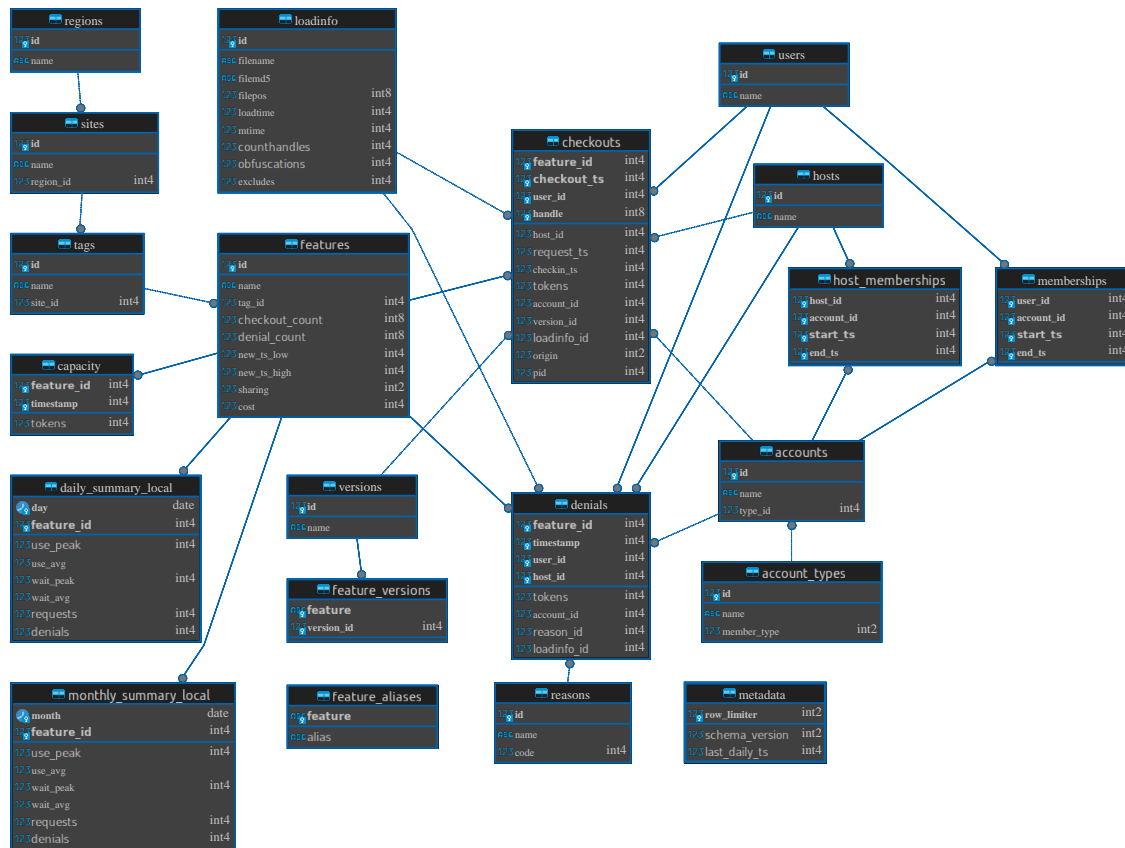


Figure 12: The Monitor Database Schema

Configuring Monitoring

General Monitoring Configuration

Adjust Global Defaults

The `vovlmd` daemon allows for the configuration of general aspects of the monitoring process via the `licmon.swd/vovlmd/config.tcl` file. In most cases, the default settings are suitable for a normal operating environment. The configurable settings, along with their default values, are described below:

```
#####  
### MONITORING CONFIGURATION ###  
#####  
  
### Period at which vovlmd will detect and apply configuration changes  
# set VOVLM(refresh) 30s  
  
### Number of seconds as the threshold of elapsed time without an update at which  
# point a license server is to be considered down (default 10 minutes)  
# set VOVLM(downthreshold) 10m  
### check frequency for the daemons (default 5 minutes)  
# set VOVLM(checkfrequency) 5m  
  
### Job periods  
# Jobs cannot run more frequently than the minimum, less frequently than the  
# maximum, and will be stopped if they exceed the autokill time.  
# Alerts are generated for jobs that run less frequently than the maximum  
# Additionally, jobs that run longer than the maximum are killed and rescheduled  
  
### Rate to run jobs that collect usage information  
# set VOVLM(period,stat,min) 30s  
# set VOVLM(period,stat,max) 10m  
# set VOVLM(period,stat,autokill) 5m  
  
### Rate to run jobs that collect expiration date information (FLEXlm and LUM)  
# set VOVLM(period,info,min) 12h  
# set VOVLM(period,info,max) 24h  
# set VOVLM(period,info,autokill) 30m  
  
### Rate to run remote LicenseMonitor sampling jobs  
# set VOVLM(period,remote,min) 5m  
# set VOVLM(period,remote,max) 30m  
# set VOVLM(period,remote,autokill) 15m  
  
### Rate to run FLEXlm debug log parsing jobs  
# set VOVLM(period,debug,min) 12h  
# set VOVLM(period,debug,max) 24h  
# set VOVLM(period,debug,autokill) 2h  
  
### Rate to run LUM denial sampling jobs  
# set VOVLM(period,lumdenial,min) 5m  
# set VOVLM(period,lumdenial,max) 30m  
# set VOVLM(period,lumdenial,autokill) 15m  
  
### Enable (1) or disable (0) splitting of multiple daemons found in the same tag
```

```
# set VOVLN(split,default) 1

### Process monitoring options
# Number of seconds to wait for processes to be removed if monitor agent is killed
# set VOVLN(processmonitor,timeout) 120
```

The above configuration establishes global defaults for all monitors. Per-tag settings can also be established in the configuration or via the web UI. See the next section for details.

Adjust Parser Settings

In addition to the global default settings shown above, parser-specific options can also be specified via the `licmon.swd/config/parser.cfg` file. The configurable settings, along with their default values, are described below:

```
# Parser configuration file.
# Should be placed in licmon.swd/config.

#####
### GLOBAL ###
#####

### Include or exclude features and subfeatures (checkouts and denials). With
includes, all
### features except those specified are ignored. With excludes, all specified
### features are ignored. Default is to allow all features.
# set PARSE(include,tag1) { feature1 feature2 featureN }
# set PARSE(exclude,tag2) { feature3 feature4 featureN }
# set PARSE(exclude,tag3) { feature5:subfeature1 feature4:subfeature2 featureN }

#####
### ALERTS ###
#####

### Set number of consecutive alert condition occurrences required to throw an
### alert. Supported alert types are:
###   allDaemonsDownAlert cannotConnectAlert clockInFutureAlert daemonDownAlert
###   droppedFeatureAlert failedToRunAlert failedToStartAlert
###   featureExpiredAlert logNotFoundAlert masterHostChangedAlert
###   noFeaturesFoundAlert scanExpirationAlert serverQuorumErrorAlert
###   serverQuorumWarningAlert tooManyMastersAlert cannotGetUsersAlert
### Default for all alerts is 3.
# set PARSE(droppedFeatureAlert) 3

#####
### FEATURE PARSING ###
#####

### Track licenses that are host or user locked. This setting causes a separate
### feature to be created that represents the locked feature (FEATURE:LOCK). As
### a result, capacity and expiration information must be obtained from the
### license file instead of the status command output. See docs for details.
# set PARSE(TAG,FEATURE,trackLockedLicenses) 1

#####
### USER PARSING ###
#####

### Override user as user@host.
# set PARSE(userAtHost) 1
```

```
#####
### HOST PARSING ###
#####

### Host name replacement based on case-sensitive regular expression pattern.
# set PARSER(hostReplace) {{PATTERN}} {REPLACEMENT}}

#####
### DENIALS ###
#####

### Throw alert if a debug log that is configured to be parsed does not exist.
# set PARSER(missingLogAlert) 1

### Define features that are to be considered equivalent to each other with
### respect to licensing. If "feature1" is denied, but a checkout of "feature2"
### immediately follows the denial, ignore the denial.
# defineAlternativeFeatures "feature1" "feature2"

### Define false denial time window. Denials for the same feature/user/host
### combination that occur within this window will be merged into one denial.
### This setting is defined in seconds. Set to 0 to only consider same-second
### denials as false. Default is 10.
# set PARSER(falseDenialWindow) 10
```

Locked Licenses

Monitor has the ability to separately track host and user-locked FlexNet Publisher licenses. If configured, locked license features will be detected in the Imstat output and processed separately from their floating counterparts. For example, the following lines in the Imstat output:

```
"MATLAB" v28, vendor: MLM
nodelocked license, locked to "ID=123456"
```

will result in a feature named MATLAB:ID=123456 being created. A prerequisite for this capability is that the capacity and expiration information be provided by the license file directly being that Imstat does not provide this information separately between floating and locked licenses.

To enable locked license tracking and indicate that the capacity and expiration information is to be obtained from the license file as opposed to the Imstat output, the parser configuration (see above) must be modified. If this is not done, only the floating license feature will be displayed and all locked license capacity and usage will be merged into that feature. To enable locked license tracking for the Matlab example above, the setting would be:

```
set PARSER(MATWORKS,MATLAB,trackLockedLicenses) 1
```

Once the setting is in place, a utility can be used to read the license file and parse the information contained therein.

```
ftlm_parse_flexlmlic: Usage Message

SYNOPSIS:
    ftlm_parse_flexlmlic [OPTIONS] TAG FILETYPE FILE

OPTIONS:
    -h                -- Print help message.
```

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

NOTE:
On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:
% ftlm_parse_flexlmlic SNPS license.dat

Each time the license file is changed, the Monitor administrator will need to run the utility to ensure that the information stored for capacity and expirations are in line with what is found in the license file.

Case Sensitivity

Monitor is designed to be case-sensitive in all areas. This means that a person whose user name is displayed as "Bob" in the license server status command output (or debug log) will be tracked independently from a person whose user name is "bob". The same is true for host names that are reported. For UNIX-only, or mixed-platform environments, this is most often desired. In some environments, particularly in environments where Windows is used as the exclusive operating system, user names may need to be treated as case insensitive ("Bob" = "bob"). A configuration item is provided to allow the administrator to force all detected user and/or host names to be lower-case. This setting is specified in the `licmon.swd/policy.tcl` file:

```
# Force tracked user names to lower case
set config(checkoutUserLowerCase) 0

# Force tracked host names to lower case
set config(checkoutHostLowerCase) 0
```

Once the change is made, the file will need to be read-in to activate the configuration change. This can be done via the "reset" function, which can be found on the **Admin > System** page, or by running the following commands at the CLI:

```
% vovproject enable licmon
% vovproject reread
```

Ensure Visibility for Tags Not Actively Being Monitored

By default, tags that are not actively being monitored via sampling are hidden from the user interface. There may be cases where visibility is still desired though. Such cases would be:

- A tag for a license server that no longer exists but reporting capability is still desired.
- A tag for a FlexNet Publisher license server that is not being sampled, but instead is being populated via a debug log.
- A tag for an Altium license server, which is solely populated via a log.

To ensure visibility for tags that fall under these conditions, use the `setTagAccess` configuration procedure as defined in [Security](#).

License Server Monitoring

Each license server to be monitored must be identified in the Monitor configuration. This can be accomplished by using the Admin web page or by editing the `vovlmd` configuration file (`licmon.swd/vovlmd/config.tcl`). The web page maintains a separate configuration, so when using both methods, be careful to configure only one monitor for each tag to avoid configuration conflicts.

Choose a Tag Name

Monitor tags are used to give symbolic names to license servers, and to distinguish between the same feature from different servers, e.g. WAN vs. LAN licenses. Tag names may contain letters, digits, underscore (`_`), and dash (`-`) characters (regexp `[a-zA-Z0-9_-]+`) and are case-sensitive.

When initially configuring Monitor, it is important to choose tag names carefully so they will not need to be changed later. Changing is possible, (see [Tag Administration](#)), but it is better to avoid this process by choosing appropriate tag names during the initial configuration phase.

One naming convention that works well is to construct tag names in the form `<site>_<vendor>_<type>`, where type is e.g. LAN, WAN, since it permits using Monitors filters to easily find things by site, tool vendor, and license type (for example `NY_MENTOR_LAN`, `SF_MENTOR_WAN`).

Browser-based Configuration

To configure monitoring via the web interface, click on the **Admin** tab and navigate to the **Monitors** page. Fill in the form with the required information for each license server and click **Add New Monitor**. Existing monitors are shown at the bottom of the page. The Edit button can be used to change the properties of an existing monitor.

Monitor Administration

Add a new license monitor using this form or edit/delete existing monitors below.

Add a new license monitor

Select monitor type: FLEXlm

Specify a unique tag name:
The tag is a label used to identify a license source that is being monitored. Use alphanumeric, underscores, and dashes only. Examples: SNPS_US, MGC_FR, CDN67

Enter license file location (port@host):
You can also use the full path to the license file.

Enter full path to status command:
On Unix-based platforms, remote servers can be queried more efficiently using a remote shell command, for example: `ssh remotehost /usr/local/bin/lmstat`. On Windows, use `/` for the path delimiter and `//` before each space if there are spaces in the path.

Path to debug log:
Optional, but required for denial tracking. Path should be specified as seen by the Altair Monitor server. Use the `@DATE@` specifier if the log is rotated daily. Use `@LATEST@` to match the latest file only. For either keyword to work, the file must contain the date in `YYYYMMDD` format. Triad debug logs must be specified directly in the `vovlmd` configuration file (`vovlmd/config.tcl`). On Windows, use `/` for the path delimiter and `//` before each space if there are spaces in the path.

Debug log time zone:
For debug log parsing, specify the time zone in which the debug log was generated if it differs from that of the Altair Monitor server. Example: `PST8PDT`.

Options:

- ☐ Prevent splitting of server list into separate tags.
- ☐ Track subfeatures in `lmstat` output.
- ☐ Track `tty/display` field in `lmstat` output.
- ☒ Track reservations shown in `lmstat` output.
- ☐ If parsing a debug log, load checkouts in addition to denials.

Figure 13: Add a Monitor

The Monitor configurations managed by the web interface are not stored in the main `vovlmd` configuration file, but are instead written to the `licmon.swd/vovlmd/config_aux.tcl` file. This file is loaded after the main configuration file and can override it.

File-based Configuration

In addition to the browser-based setup, you may specify license servers to be monitored by editing the `.../licmon.swd/vovlmd/config.tcl` file. If the file is not already present, you may copy a complete and self-documented example from `$VOVDIR/etc/config/vovlmd/config.tcl`.

Supported License Managers

The following table shows licensing systems that are supported in this release, and has links to the reference documentation for the configuration options that are available for each system.

System	Config Statement
Altair	add_ALTAIR
Altium Log File	add_ALTIM_LOG
BETA CAE BETA LM	add_BETA
ClearCase	add_CLEARCASE

System	Config Statement
Wibu CodeMeter Log File	add_CODEMETER_LOG
Dassault DSLS	add_DSLS
Dassault DSLS Log File	add_DSLS_LOG
Green Hills Legacy (Elan)	add_GHS_LEGACY
FlexNet Publisher	add_LM_LICENSE_FILE
FlexNet Publisher Log File	add_DEBUG_LOG
FlexNet Publisher	add_LIC_OPTIONS
Fujitsu ICAD (Windows only)	add_ICAD
GNS	add_GNS
Green Hills	add_GHS
IBM LUM	add_LUM
LSTC	add_LSTC
Wolfram MathLM	add_MATHLM
Remote Monitor	add_REMOTE_LM
Reprise RLM	ad_REPRISE
Altair License Key File (RTDAKEY)	add_RTDAKEY
Sentinel HASP	add_HASP
Sentinel RMS	add_SENTINEL
Silvaco	add_SILVACO
OLicense	add_OLICENSE
QF-Test	add_QFTEST
T-Systems LICMAN	add_TSYSTEMS
X-Formation LM-X	add_LMX
Process Tracking	add_PROCESS_TRACKING

Configuration Example

```
add_LM_LICENSE_FILE 5280@lichost1 -tag CADENCE
add_LM_LICENSE_FILE 2233@lichost1,2233@lichost2,2233@lichost3 -tag SYNOPSYS
add_LM_LICENSE_FILE /vendor/mentor/current/license/license.dat -tag MENTOR
add_LM_LICENSE_FILE 1717@lichost4 -tag MGC -lmstat /tools/flexlm/bin/lmstat
```

About FlexNet Publisher Reservations

In both configuration methods, there is an option to enable the tracking of FlexNet Publisher reservations. Reserved licenses that are not used are considered checkouts by Monitor if this option is enabled. The checkouts are for a special user, beginning with "r:", that includes the reservation object name, so that these checkouts can be distinguished from real usage in the Monitor reports. Enabling this option allows Monitor to quickly and easily identify waste that is created as a result of obsolete and overly aggressive reservations.

Status Command Specification

Before Monitor can obtain data from any license server, a supported status command must be present to interface with. Some status commands are simple in their installation and operation, requiring only a single binary to query a license server (eg FlexNet Publisher `lmstat`, Reprise's `rlmstat`, and Sentinel's `lsmon`). Others, however, require a full installation and configuration (eg ClearCase and IBM's LUM).



Important: It is important to understand that unless a status command is present and working on the Monitor server machine outside of the Monitor environment, no data will be obtained.

It is recommended to test the status command that is to be used by Monitor manually with a UNIX shell or Windows command prompt to validate its operation before configuring Monitor to use it.

Command Versions

By default, Monitor searches the system path for the status command. It is common practice to copy the status commands to the `$VOVDIR/bin` directory so that there is at least a default command always available. However, due to version incompatibilities, this may not always work. For this reason, it is recommended to point to the version of status command that matches that of the license server being monitored.

If a specific status command is defined in the configuration, the path to this command is honored. This is accomplished in the browser interface by filling in the status command form input with a full path to the desired status command. In the configuration file, each license type has a status command argument that is used to specify the location. For example, the `-lmstat` argument is used to specify a path to the `lmstat` command for FlexNet Publisher-based monitors.



Note:

- **Sentinel:** Some versions of Sentinel's `lsmon` utility on Windows have the behavior of requiring the enter key to be pressed at the end of its output dump. This means that when executed as a background task in Monitor, the task will never end. If the `lsmon` utility being interfaced with exhibits this behavior, make a copy of the `vovlsmon.bat` script template found in `$VOVDIR/bat` and place it somewhere outside of the `$VOVDIR` directory. Modify it to point to the `lsmon.exe` being interfaced with, then configure the status command to point to this script instead of `lsmon.exe` directly.

Working With Remote License Servers

For UNIX-based installations, the status command can be used in conjunction with `rsh` or `ssh` to increase the performance of monitoring license servers across the WAN. FlexNet Publisher uses small packets, leading to many round-trips to get all of the required information. Remote execution of `lmstat` on the remote license server (or on a machine located in the same location as the license server) via `rsh/ssh` can speed up the response significantly when compared to executing `lmstat` on the Monitor host against a remote license server. If the remote server is located in a different time zone, the time zone should be overridden to the time zone in which the Monitor server is located so that reported checkout times will be tracked correctly. See below for some examples of using `ssh` in the status command specification:

```
ssh remotesrv /usr/local/bin/lmstat  
ssh remotesrv env TZ=PST8PDT /usr/local/bin/lmstat
```

For this to work, you must have an account at the remote site that can remote-shell without a password, by using `rsh` or setting up `ssh` to use authorized keys.

Debug Log Monitoring

License utilization data that Monitor gathers primarily comes from license management tool's status command (e.g. the FlexNet Publisher `lmstat` command).

For FlexNet Publisher and DSLS, the status command does not provide denial information, so the debug log must be analyzed to obtain that information. For the Altium and CodeMeter license manager, there is no status command at all, so its debug log must be used for usage data. The CodeMeter debug log does not contain denial information.

Enable FlexNet Publisher Debug Logs

This section describes how to enable FlexNet Publisher debug logs. It is crucial to understand the differences found in the debug log based upon the enable method. These differences will dictate how Monitor will be configured to import data from the logs.

To enable the debug log in FlexNet Publisher, use one of the following methods.

Static Debug Log

A static debug log is one that captures both the `lmgrd` and vendor daemon output. The `lmgrd` daemon contributes date stamps, while the vendor daemon contributes all license activity.

At license server startup time, pass the `-l </path/to/debug/log>` option to `lmgrd`. This will create a debug log that will contain all of the information necessary for Monitor to properly analyze the log contents. However, this file is not rotatable, and will require a restart of the server to rotate the log. This type of log will grow over time, the rate at which is determined by the amount of activity on the license server. If disk space is a concern, the rotating method below should be considered.

Rotating Debug Log

A rotating debug log is one that captures only the vendor daemon output, which means there will be no date stamps in the file. Because of this, there is a chance for inaccurate assumptions in the log. For example, if license activity ceases for more than 24 hours, the day will become unknown due to there not being a visible roll-over at midnight. To address this, the log must be rotated daily, typically a few minutes after midnight.

- In the options file for FlexNet Publisher, use the `DEBUGLOG </path/to/debug/log>` directive. So that Monitor can appropriately handle FlexNet Publisher server restarts, the debug log name specified in this directive must be in the format of `<file>.startup`. This will indicate that a restart has occurred.
- In a triad environment, the `DEBUGLOG` directive may specify the same physical file since the MASTER server in the triad is the only one that writes to the log file at any given time. The log files may also be kept separate for each server as long as they are all specified in the Monitor configuration.

In this method, the debug log must be rotated daily. This will ensure that the license activity in the log is always for the current 24-hour period and will also keep the file size small. Monitor will need to be configured to automatically look for a daily log file. For this to work properly, the debug log must be rotated to a file with a name reflecting the current date, in the format of `<fileName>.<YYYYMMDD>`, using the `lmswitch` command in conjunction with a task scheduling facility.

For UNIX, use *cron*

Example cron task:

```
# FlexNet Publisher debug log cron tasks for Monitor.
# Make sure to keep each cron job on one line and either use full paths for each
# file
# or change into the directory where the file exists.
# To suppress cron mail from being sent, add > /dev/null 2>&1 after each
# command.
## If the server is restarted for any reason, switch the debug log to the
# current day
# so it can be parsed, then remove the startup log.
# LINE SPLIT IN THIS EXAMPLE: SHOULD NOT BE SPLIT IN THE REAL CONFIG FILE.
* * * * * [ -f debug.startup ] &&    lmswitch -c license.dat vendord debug.`date
+ \%Y\%m\%d`.log && rm -f
debug.startup
## Rotate the log every night at midnight
0 0 * * * lmswitch -c license.dat vendord debug.`date +%Y%m%d`.log
```

Using cron is the most reliable approach to performing the rotation successfully. The first cron task will detect if a FlexNet Publisher server restart has occurred via the presence of the `.startup`

debug log. If this log exists, the debug log will be switched to the standard .YYYYMMDD log name format, which is automatically parsed by Monitor. This task runs every minute so that gaps in the data collection are kept at a minimum. The second task will switch the log file every night at midnight to a name that matches the current day, leaving the previous day's log behind for archive purposes.

For Windows, use the Task Scheduler:

To perform rotation on Windows, the Task Scheduler application must be used in conjunction with a batch script as follows:

- Use the following example to create a batch script (modify the rotation time to the desired time).
- Create a scheduled task in Task Scheduler by pointing to the batch script and selecting it as the application to run. Set it to run every day.
- After the task is created, open the Advanced Properties for the task and set it to repeat every one minute. The batch script itself will make sure the appropriate action is taken at the right time.

Example batch file:

```
@echo off

REM FlexNet Publisher debug log rotation script for Monitor.
REM Make sure to use full paths for each file or change
REM into the directory where the file exists.

REM Define rotation time (24-hour clock)
set rotate=00:00

REM Get today in YYYY/MM/DD format
set today=%date:~10,4%%date:~4,2%%date:~7,2%

REM Get time in HH:MM format
set now=%time:~0,5%

REM If the debug.startup is present, the server has been restarted
REM and the log should be renamed so Monitor will parse it
if exist debug.startup goto STARTUP

REM If the script is called at the rotation time, perform the rotation
if %now% == %rotate% goto ROTATE
goto DONE

:STARTUP
lmswitch -c license.dat vendord debug.%today%.log
del debug.startup
goto DONE

:ROTATE
lmswitch -c license.dat vendord debug.%today%.log
goto DONE

:DONE
```

Once the debug log has been configured and is being generated by the license manager, Monitor must be configured to parse the log. This can be done jointly with a sampling monitor, or a debug log can be monitored in a stand-alone fashion.

Joint Debug Log Monitoring (FlexNet Publisher only)

Web-based Configuration

The form in the monitor administration web page contains a field for specifying the path to a single debug log for FlexNet Publisher servers being configured. When multiple debug logs exist for a single tag, the file-based stand-alone configuration must be used (see below).

Monitor Administration

Add a new license monitor using this form or edit/delete existing monitors below.

Add a new license monitor

Select monitor type:

FLEXlm

Specify a unique tag name:

The tag is a label used to identify a license source that is being monitored. Use alphanumeric, underscores, and dashes only. Examples: SNPS_US, MGC_FR, CDN67

Enter license file location (port@host):

You can also use the full path to the license file.

Enter full path to status command:

On Unix-based platforms, remote servers can be queried more efficiently using a remote shell command, for example: `ssh remotehost /usr/local/bin/lmstat`. On Windows, use `/` for the path delimiter and `//` before each space if there are spaces in the path.

lmstat

Path to debug log:

Optional, but required for denial tracking. Path should be specified as seen by the Altair Monitor server. Use the `@DATE@` specifier if the log is rotated daily. Use `@LATEST@` to match the latest file only. For either keyword to work, the file must contain the date in `YYYYMMDD` format. Triad debug logs must be specified directly in the `vovlmd` configuration file (`vovlmd/config.tcl`). On Windows, use `/` for the path delimiter and `//` before each space if there are spaces in the path.

Debug log time zone:

For debug log parsing, specify the time zone in which the debug log was generated if it differs from that of the Altair Monitor server. Example: `PST8PDT`.

Options:

☐ Prevent splitting of server list into separate tags.

☐ Track subfeatures in `lmstat` output.

☐ Track `tty/display` field in `lmstat` output.

☒ Track reservations shown in `lmstat` output.

☐ If parsing a debug log, load checkouts in addition to denials.

Figure 14: Add a Monitor

Optionally, checkout information can be acquired in addition to denial information by checking the box for this functionality. This feature is currently considered experimental as Altair Accelerator continues to improve the handling of corrupt and incomplete data commonly encountered with debug logs.

File-based Configuration

In the `add LM_LICENSE_FILE` configuration procedure, the `-debuglog` option allows for a file path to be specified that will create a debuglog parsing job that will run for the tag specified in the configuration, in addition to the sampling job. When multiple debug logs exist for a single tag, the file-based stand-alone configuration must be used.

Optionally, checkout information can be acquired in addition to denial information by adding the `-debuglogcheckouts` option. This feature is currently considered experimental as Altair Accelerator continues to improve the handling of corrupt and incomplete data commonly encountered in debug logs.

Stand-Alone Debug Log Monitoring

There may be situations where a debug log will need to be parsed in a stand-alone fashion as opposed to directly coupling it to a sampling configuration. In most cases, this will be any time there are multiple debug logs for the same tag.

The following lists some of the situations that require a stand-alone debug log configuration:

- FlexNet Publisher and DSLS triad configurations, where a debug log is generated for each server in the triad.
- FlexNet Publisher multi-daemon license files, where a debug log is generated for each daemon found in the license file.
- Secure areas where sampling is not permitted.
- Altium and CodeMeter, since there is no status command to sample.

Unlike the joint monitoring configuration, the stand-alone configuration method allows for binding more than one debug log to any tag that is also being monitored for usage. This can be accomplished via file-based configuration only.

Debug Log Configuration

Web-based Configuration

The form in the Monitor administration web page contains a DebugLog monitor type for FlexNet Publisher, an AltiumLog monitor type for the Altium License Manager and a CodeMeterLog monitor type for the CodeMeter License Manager. These are used to specify a single debug log for a specific tag:

The screenshot shows a web form titled "Monitor Administration" with the subtitle "Add a new license monitor using this form or edit/delete existing monitors below." The form has a section "Add a new license monitor" with the following fields:

- Select monitor type:** A dropdown menu with "FLEXlm Debug Log File" selected.
- Specify a unique tag name:** A text input field. Below it is a note: "The tag is a label used to identify a license source that is being monitored. Use alphanumeric, underscores, and dashes only. Examples: SNPS_US, MGC_FR, CDN67".
- Path to debug log:** A text input field. Below it is a note: "Optional, but required for denial tracking. Path should be specified as seen by the Altair Monitor server. Use the @DATE@ specifier if the log is rotated daily. Use @LATEST@ to match the latest file only. For either keyword to work, the file must contain the date in YYYYMMDD format. Triad debug logs must be specified directly in the vovlmd configuration file (vovlmd/config.tcl). On Windows, use / for the path delimiter and // before each space if there are spaces in the path."
- Time zone:** A text input field. Below it is a note: "For debug log parsing, specify the time zone in which the debug log was generated if it differs from that of the Altair Monitor server. Example: PST8PDT."

Figure 15: Stand-alone Monitoring

For situations where multiple debug logs exist for a single tag, use the file-based configuration method below.

Optionally, for FlexNet Publisher, usage information can be acquired in addition to denial information by checking the box for this functionality. This feature is currently considered experimental as Altair Accelerator continues to improve the handling of corrupt and incomplete data commonly found in debug logs. Altium and CodeMeter debug logs only contain usage information, so denials are not supported.

File-based Configuration

To enable stand-alone FlexNet Publisher or DSLS debug log parsing, use one of the following configuration procedures:

- [add_DEBUG_LOG](#)
- [add_DSLS_LOG](#)

These procedures must be written into the `licmon.swd/vovlmd/config.tcl` file. Repeat for specifying multiple debug logs for triads. Supply the license server host name that generated each log using the `-host` argument. Also repeat for specifying multiple debug logs that are the result of multiple daemons being served on the same tag that each have their own debug log.

Optionally, checkout information can be acquired in addition to denial information by adding the `-checkouts` option. This feature is currently considered experimental as Altair Accelerator continues to improve the handling of corrupt and incomplete data commonly found in debug logs.

For Altium usage monitoring, use the [add_ALTIM_LOG](#) procedure in the `licmon.swd/vovlmd/config.tcl` file.

For CodeMeter usage monitoring, use the [add_CODEMETER_LOG](#) procedure in the `licmon.swd/vovlmd/config.tcl` file.

Debug Log Specification

For both the joint and stand-alone monitoring methods described above, the debug log can be specified in the following formats:

- `/path/to/file` (for static logs)
- `/path/to/file.@DATE@.log` (for rotated logs)
- `/path/to/file.@LATEST@.log` (for rotated logs)

The first option is a simple path to the debug log and is typically used when a static debug log is being parsed. As stated above, a static debug log is not rotated on any basis and normally contains data for multiple days. In order to be parsed correctly, the file must contain `TIMESTAMP` lines. The file can be named something as simple as `debug.log`, in which case data tracking will begin upon the first `TIMESTAMP` line encountered. This ensures that the records will be applied to the correct day and time. The filename can also contain the start date for the data, such as `debug.20100101.log`. If a date is found in the filename in the format of `YYYYMMDD`, it will be used as the initial day and time instead of requiring the parser to find it in the debug log data itself. The drawback to using a dated filename with the static method is that the monitor configuration must be updated if the debug log file name is changed.

The second option utilizes the `@DATE@keyword`, which is substituted with the current date, in `YYYYMMDD` format, at parsing time. This is required when the debug log is created using the rotation method. The rotation method results in a debug log for each day, named in the aforementioned format. Rotation is required if the debug log does not contain `TIMESTAMP` lines.

The third option utilizes the `@LATEST@keyword`, which causes the parser to search for the latest file found based on its name. The file name must contain its creation date, in `YYYYMMDD` format. This is required mainly for Altium logs, which are rotated at a static rate that is less than daily.

Time Zones

When monitoring a debug log that was written by a license server located in a time zone that differs from the time zone in which the Monitor server is running, the remote time zone must be properly specified. Otherwise, denial events will not be synchronized with the capacity and utilization data that is coming from sampling. The remote time zone can be specified with both the web interface and the configuration file, via the `-tz` argument to `add_LM_LICENSE_FILE`, `add_DEBUG_LOG`, and `add_ALTIIUM_LOG`.

Example time zone specifications can be found in [Time Zones](#).

Live Monitoring and Manual Parsing

Once a debug log has been specified for a tag, Monitor will parse the log every 12 hours, by default. This period can be adjusted in the **Admin > System > Configuration Information** page or via the associated configuration procedure in the `licmon.swd/vovlmd/config.tcl` file.

If it is determined that the debug log environment is not suitable for live monitoring, logs can be manually parsed using either the appropriate parsing command for each license manager type:

- [FlexNet Publisher Debug Log](#)
- [DSLS debug logs](#)
- [Altium debug logs](#)
- [CodeMeter debug logs](#)

Usage Information in FlexNet Publisher Debug Logs

Although experimental support is included in Monitor for parsing debug logs for utilization data, it is strongly discouraged to rely upon debug logs for the main data source for utilization statistics. The following lists some reasons for this:

- Data is often missing from the debug log, meaning that there are missing OUT and/or IN records. It only takes a few missing records to produce inaccurate results, especially for long checkouts.
- The OUT and IN records cannot be precisely matched to one another because they do not have unique identifiers like `lmstat` provides (the FlexNet Publisher handle). This makes it impossible to accurately track checkout durations. The Monitor debug log parser uses a first-out, last-in algorithm. This allows for easier detection of erroneous records (outliers) in the `ftlm_check_db` utility.
- Debug logs are often corrupt with malformed lines. These are discarded by the parser, but they were supposed to represent a record that should be tracked, but it is lost because of the corruption.
- The license server clock is often adjusted by a clock synchronization daemon such as `ntpd`. When this occurs, the timestamps in the debug log can jump forward or backward. In some cases, this the jump is significant enough to cause major skew in the data. The parser attempts to accommodate for this, but if the jump is large enough in the backward direction, it will ignore records until the clock catches back up with the last known good timestamp.

False Denial Filtering

If denial monitoring has been enabled, historical denial reports will be available. Denials are obtained from the debug log of the license daemon and are filtered when parsed so that false denials are not loaded into the Monitor database and included in reports.

Denials are recorded in a debug log with an event label, such as the "DENIED" label in FlexNet Publisher. The denied label in the debug log cannot be relied upon to always indicate a true denial of a license checkout. There are many situations that can create false denials.

Monitor handles the following false denial situations.

False Denial Example 1: Denial Followed by Denial (same second)

This example shows two denials that occurred in the same second:

```
14:59:00 (xyzd) DENIED: "Affirma" jas@cs2 (Licensed number of users already
reached(-4,342))
14:59:00 (xyzd) DENIED: "Affirma" jas@cs2 (Licensed number of users already
reached(-4,342))
```

The first denial is filtered as false because within the same second that another denial occurred.

False Denial Example 2: Denial Followed by Denial (window)

This example shows two denials that occurred within a 10-second window of one another:

```
14:59:00 (xyzd) DENIED: "Affirma" jas@cs2 (Licensed number of users already
reached(-4,342))
14:59:09 (xyzd) DENIED: "Affirma" jas@cs2 (Licensed number of users already
reached(-4,342))
```

The first denial is filtered as false because within the 10-second window that another denial occurred. The window is configurable (default: 10 seconds). Refer to [General Monitoring Configuration](#) for details on configuring the window.

False Denial Example 3: Denial Followed by Checkout

This example shows multiple denials, followed by an immediate checkout:

```
14:59:00 (xyzd) DENIED: "Affirma" jas@cs2 (Licensed number of users already
reached(-4,342))
14:59:01 (xyzd) OUT: "Affirma" jas@cs2
```

This is filtered as a false denial because a checkout was obtained within 1 second that denial event occurred.

False Denial Example 4: Equivalent Features

Some tools can utilize more than one license feature to satisfy its licensing requirement. These tools will have some hard-coded order in which they attempt checkouts, and will continue to search through the available features until it finds one that is available for checkout:

```
23:14:24 (xyzd) DENIED: "Affirma" edl@cs82 (Licensed number of users already
reached(-4,342))
```

```
23:14:24 (xyzd) DENIED: "Affirma" edl@cs82 (Licensed number of users already
reached(-4,342))
23:14:25 (xyzd) DENIED: "Affirma" edl@cs82 (Licensed number of users already
reached(-4,342))
23:14:25 (xyzd) DENIED: "Affirma" edl@cs82 (Licensed number of users already
reached(-4,342))
23:14:26 (xyzd) OUT: "MM_Sim" edl@cs82
```

In this example, the tool first tries to check out an "Affirma" license but then falls back on a "MM_Sim" license within 1 second. From the point of view of the tool, there is no denial, and thus Monitor can be instructed to not consider it one. To eliminate these denials, include the following statement in the `licmon.swd/config/parser.cfg` file:

```
defineAlternativeFeatures "MM_Sim" "Affirma"
```

If the file does not exist, an example file exists at `$VOVDIR/etc/config/lm/parser.cfg` that can be copied, or the file may be created by hand.

Filter Improvements

Altair relies on customer feedback to determine the effectiveness and areas of improvement for denial filtering. Engineering will modify the heuristics of the filters as new false denial situations are reported to [Altair Engineering Support](#).

Monitor Support

Currently, Monitor supports parsing FlexNet Publisher, Dassault DSLS, and Altium debug logs for denial information. CodeMeter debug logs do not contain denial information. Please contact [Altair Engineering Support](#) to request support for additional license managers if needed.

Multi-Site Support

Companies with multiple sites may decide to implement Monitor at each site instead of monitoring remote license servers across the WAN. Monitor supports two methods of aggregating remote data to a "main" site.

The first method is a sampling method that queries the remote Monitor servers via HTTP. This provides current metrics in the main Monitor site for each remote server being monitored, albeit with a low sampling rate. This approach is good for those who would like to monitor remote sites for license server health and have a high-level view of the usage as well. However, because the sampling rate is low, the granularity of the data collected may not be suitable for running accurate historical reports. Denial data is not supported in this configuration either. If historical report accuracy and/or denial data is of importance, the second method should be considered.

The second method does not sample the remote instances, but instead transfers and loads the remote data files into the main site's database. While this does not provide server health and current usage monitoring, it does provide historical reporting capabilities at the maximum granularity possible.



Note: Currently, it is not possible to employ both aggregation methods due to conflicts that would be created in the main site database.

Method 1: Sampling

Web-based Configuration

The form in the monitor administration web page contains a monitor type for specifying a remote Altair Monitor instance to monitor. The site that is specified will be automatically prepended to the tag so that it will be unique in the main site.

File-based Configuration

To monitor a remote Monitor instance, use `add_REMOTE_LM` in the `licmon.swd/vovlmd/config.tcl` file. When using this method, make sure that the tag name is unique and indicates from what site the data originates.

Method 2: Data Transfer

To use this method, there must not be any tag name conflicts between any of the remote sites and the main site. Because of this, it is recommended to include a site identifier in the each tag name. For example, for a main site located in New York, there may be an "EDA_NY" tag to represent the "EDA" license server. If a remote site in San Francisco also has an "EDA" license server, it should be named similar to "EDA_SF". This will allow for both individual and joint reporting in the main site.

To transfer checkout, denial, and capacity data from the remote site to the main site, Monitor relies upon the operating system's task scheduling capability. Once the transfer task is put into place on the remote Monitor server, it will transfer data to the main site per the schedule specified by the administrator. Once the data is transferred, a periodic job is used to load the data automatically into the main site's database.

UNIX Configuration

On UNIX platforms, the cron facility is utilized. The example below uses the UNIX `rsync` utility specifically, but other file delta-based copy/sync utilities will work as well. The example uses a weekly schedule to perform the data transfer, beginning at 2am every Saturday night. All checkouts data files are transferred into a subdirectory that represents that site from which it originates. The remote site is in San Francisco and the main site is in New York. The main site server name is "nylmsrv".

```
# CHECKOUT DATA
# If main site licmon.swd is available via NFS
0 2 * * 7 /bin/rsync -az --delete /path/to/SF/licmon.swd/data/checkouts/
*.chk.gz \
    /path/to/NY/licmon.swd/data/checkouts/SF

# If main site licmon.swd is not available via NFS
# Use as a guide for denial and capacity data as well
0 2 * * 7 /bin/rsync -az --delete /path/to/SF/licmon.swd/data/checkouts/
*.chk.gz \
    nylmsrv:/path/to/NY/licmon.swd/data/checkouts/SF

# DENIAL DATA
0 2 * * 7 /bin/rsync -az --delete /path/to/SF/licmon.swd/data/denials/* \
    /path/to/NY/licmon.swd/data/denials/SF
```

```
# CAPACITY DATA
0 2 * * 7 /bin/rsync -az --delete /path/to/SF/licmon.swd/data/capacity/*.cap
\
/path/to/NY/licmon.swd/data/capacity/SF
```



Note: The example above wraps the configuration line for readability but cron requires each task to be defined entirely on one line.

Windows Configuration

On Windows, the Windows tasks scheduler is used to configure and execute the data transfer task. The example below uses the robocopy utility. Windows Vista, 7, and Server 2008 include this utility, which supports delta-based file copying. On Windows XP, the Windows Resource Kit must be installed in order to have access this utility. In this example, the remote site is in San Francisco and the main site is in New York. The main site server name is "nylmsrv". Using the Windows task scheduler, create a scheduled task to run at the desired schedule using a command syntax similar to:

```
# CHECKOUT DATA
# If the main site is available via persistent drive letter
robocopy C:\path\to\SF\licmon.swd\data\checkouts E:\path\to\NY\licmon.swd\data
\checkouts\SF /MIR /Z

# If the main site is not available via drive letter (use as a guide for denial
and capacity data as well)
robocopy C:\path\to\SF\licmon.swd\data\checkouts \\nylmsrv\path\to\NY\licmon.swd
\data\checkouts\SF /MIR /Z

# DENIAL DATA
robocopy C:\path\to\SF\licmon.swd\data\denials E:\path\to\NY\licmon.swd\data
\denials\SF /MIR /Z

# CAPACITY DATA
robocopy C:\path\to\SF\licmon.swd\data\capacity E:\path\to\NY\licmon.swd\data
\capacity\SF /MIR /Z
```

Automatic Data Loading

To enable automatic loading of the data that is transferred from remote sites into the main site a periodic job is available. The Configuration Information page can be used to configure this task. Refer to the documentation for the [LOAD_REMOTE_DATA](#) task for more details.

Project and Group Management

For reporting purposes, users can be grouped to keep track of projects, departments, or other organizational units.

By default, Monitor supports one group, referred to as "project", for each checkout record. The project is assigned to each checkout and shows up in all current and historical reports for checkout data. The value of the project is "none" unless otherwise configured per the instructions below. The value can also be specified at the command line by the checkout user. This allows for more granular reporting of

projects, using the LM_PROJECT environment variable, the current UNIX group, or a specified value. For more information on this capability, see [Project Tracking](#).

If grouping by organizational units other than "project" is required (for example, departments or sites), custom groups can be defined. Custom groups are comprised of time-based memberships for users. This allows for fine-grained control over users and their groups. Group memberships can be defined via the web interface, configuration file, or they can be driven by LDAP. See below for more details on defining custom groups and memberships.



Note: Project assignments are currently not supported for denial data. Custom group memberships apply for all data.

Web-based Project Configuration

To define project assignments via the web interface, click on the **Admin** tab and navigate to the **Groups** page:

Group Management

Group users or hosts into projects or custom groups for reporting purposes. This page creates an account definition file that is used by the `film_accounts` utility, which can be used to set both live and historical accounts for checkouts.

Project	Members	Unix Groups
<input type="text"/>	<input type="text"/>	<input type="text"/>

- To delete a project, clear the project name and click on 'Save Configuration'.
- This page edits file: `c:/altair/swd/licmon.swd/config/accounts.web.cfg`.
- Additional definitions can be specified in: `c:/altair/swd/licmon.swd/config/accounts.cfg`, which does not currently exist.
- Each user should be assigned to only one project at a time.
- You may also optionally use a Unix group to populate the project definition.
- For users found in multiple Unix groups, the first assignment encountered is used.

Save ConfigurationApply Configuration...

Figure 16: Groups - Project

The default sub-page is for managing project assignments. Projects can be defined, along with the users that belong to them and/or associated UNIX groups (if Monitor is running on a UNIX-based platform). If using UNIX groups, and a user is found in more than one group, the first group appearance is honored.



Note: Configuring projects via the web interface only establishes the configuration. One of the two assignment methods below will need to be performed in order for the account definitions to be activated.

Web-based Custom Group Configuration

Custom accounts and types can be managed by visiting the appropriate sub-page of the accounts management page above. Each custom account must be of a certain type. To define a type, simply enter its name. To rename an existing type, select it from the list and type in the new name.

Group Management

Group users or hosts into projects or custom groups for reporting purposes. This page creates an account definition file that is used by the ftm_accounts utility, which can be used to set both live and historical accounts for checkouts.

Custom Group Types: All custom groups must be associated with a custom group type. Custom group types can be managed below.

Delete Type - or -

Rename Type

User-based

New Type

Figure 17: Groups - Custom Group Types

Once a type has been created, the type will show up in the custom group sub-page. From here you can create groups for each type that has been created.

Group Management

Group users into projects or custom groups for reporting purposes. This page creates an account definition file that is used by the ftm_accounts utility, which can be used to set both live and historical accounts for checkouts.

Custom Groups

Custom groups can be used to keep track of activity by department, location, and more. Configuration controls for existing custom groups can be found below, organized by the group type.

department:

Custodial Arts

Edit Group

Delete Group

- or -

Rename Group

New Group

division:

Engineering

Edit Group

Delete Group

- or -

Rename Group

New Group

location:

Beverly Hills

Edit Group

Delete Group

- or -

Rename Group

New Group

Figure 18: Custom Group Administration Page

Once a custom group has been created, it can be edited, deleted, or renamed. Edit the group to add or remove members, as well as select a member to edit its memberships.

Group Management

Group users or hosts into projects or custom groups for reporting purposes. This page creates an account definition file that is used by the ftim_accounts utility, which can be used to set both live and historical accounts for checkouts.

Custom Groups: Custom groups can be used to keep track of activity by department, location, and more. Configuration controls for existing custom groups can be found below, organized by the group type.

Editing user-based department group 'engineering'...

Add a new member below. Leave the end time blank for a 10-year membership.

accel

Start:

End:

Add Member

Cancel

Figure 19: Edit Custom Group

To view edit a member's memberships, select the user from the drop-down menu and choose the edit function. A form containing a list of existing memberships will appear that can be used to add, update, or delete membership periods.

Group Management

Group users or hosts into projects or custom groups for reporting purposes. This page creates an account definition file that is used by the ftim_accounts utility, which can be used to set both live and historical accounts for checkouts.

Custom Groups: Custom groups can be used to keep track of activity by department, location, and more. Configuration controls for existing custom groups can be found below, organized by the group type.

Editing user-based department group 'engineering'...

Choose an existing member to edit that member's membership for this group or add a new member using the form below.

ajoseph

Edit Member

Remove Member

Editing engineering membership for user 'ajoseph'

Update, clear, or add a new membership period for this group and user using the form below. Leave the end time blank for a 10-year membership.

Start:

05/08/2023 00:00:00

End:

08/07/2023 23:59:59

Update Period

Delete Period

Start:

06/12/2023 00:00:00


End:

08/08/2023 23:59:59

Add Period

Cancel

Figure 20: Edit Membership



Note: Configuring custom groups via the web interface only establishes the configuration. One of the two assignment methods below will need to be performed in order for the group definitions to be activated.

File-based Configuration

Both projects and custom groups can be defined via configuration file as well. This enables an administrator to keep an accurate list of assignments for all users that is easily maintained. The configuration files are:

- licmon.swd/config/accounts.cfg - Hand-written configuration file.

- `licmon.swd/config/accounts.web.cfg` - Configuration file that is generated by the group administration page, not to be hand-edited.

The administrator must take care that there are no conflicts in the assignment of users to projects and groups between the definitions found in both files. If conflicts are detected, the first assignment is honored. It is recommended to use one approach for all definitions to avoid conflicts altogether. The configuration files are used for the first two group management methods described below.

Example Configuration: `$VOVDIR/etc/config/lm/accounts.cfg`

```
#
# Static project definition
# These definitions are read by "ftlm_accounts setliveall", which is called once
# every hour by the "live_update_accounts" liveness script.
#
set FTLM_ACCOUNT(GPS_CHIP) {
    joe
    bob
    sue
}

#
# Project defined via Unix group, first group encountered for a user is honored
# These definitions are read by "ftlm_accounts setliveall", which is called once
# every hour by the "live_update_accounts" liveness script.
#
set FTLM_UNIXGROUPS(GPS_CHIP) {
    gps
}

#
# Static custom group definition
# These definitions are read by "ftlm_accounts loadfromfile", which is called
# once every hour by the "live_lm_update_memberships_from_file" liveness script,
# and can be used to define user- and host-based group types.
# These definitions establish a group membership starting at the time the file
# is read and ending 10 years later by default. Use -start and -end to specify
# other times.
#
# Time formats supported are:
#     10-digit Unix timestamp
#     YYYYMMDD#         YYYY-MM-DD
#     Full date as shown in "ftlm_accounts -showmemberships" output
#
#           TYPE           GROUP   USERLIST   OPTIONS
UserGroup "department"    "GPS" {
#           USERLIST   OPTIONS
    Member "joe"
    Member "bob mark" -start "20210101" -end "20211231"
}

HostGroup "Site"          "San Jose" {
    Member "sjhost1 sjhost2"
}

#
# Custom group definition via NIS maps
# Define NIS data to use for populating group memberships in the LM DB.
# These definitions are read by "vovnis_update_memberships", which is called
# once every hour by the "live_lm_update_memberships_from_nis" liveness script.
# These definitions establish a group membership starting at the time NIS is
```



```
# is queried and ending 10 years later. Formats are:
#
# set NIS(<customGroupName>,map)      <nisMapName>
# set NIS(<customGroupName>,column)  <columnNumber>
# set NIS(<nisMapName>,usercolumn)   <columnNumber> (default is 1 if unspecified)
#
set NIS(location,map)      sites
set NIS(location,column)  2
set NIS(group,map)        organizations
set NIS(group,column)     4
set NIS(department,map)   organizations
set NIS(department,column) 5
```

Project Configuration Activation

Once project and/or custom group definitions have been created, they must be activated in order for projects to be applied to new checkouts and for custom groups to be populated in the Monitor database.

The `ftlm_accounts` utility can be used to manually manage projects. The usage syntax for the utility is:

```
ftlm_accounts: Usage Message

Utility to show and manage user project and custom group definitions.

USAGE:

    ftlm_accounts [OPTIONS] <ACTION> ....

PROJECT ACTIONS:

    showdb                - Show ALL project assignments found for each user
                           in the database. For users with no project
                           assignments, the default of "none" is printed.
    autofill              - Automatically assign projects to users by
                           searching for previously assigned projects. The
                           most frequently used project is assigned. If
                           none are found, a warning is printed.
    setdb <users> <project> - Assign a project to one or more users. If a
                           user is not already in the database, an error is
                           printed for that user. To clear all project
                           assignments for the specified user(s), pass ""
                           as the <project>.
    reset                 - Clear all project assignments for all users.
                           WARNING: This is destructive and can take a
                           while.
    showlive              - Show project assignments that are to be applied
                           to new checkouts as they occur.
    setlive <users> <project> - Assign a default project to one or more users
                           for checkouts that will be detected by
                           Monitor.
    setliveall            - Assign a default project for all live checkouts
                           based on the definitions in the project
                           configuration.
    clearlive             - Clear all live project assignments.

PROJECT OPTIONS:

    -origin <N>           - Restrict to given origin.

CUSTOM GROUP ACTIONS:
```

addmembership	- Deprecated. Use addusermembership.
addhostmembership <type> <group> <host> <startDateSpec> <endDateSpec>	- Add a host to a host-based custom group. If overlap is detected, an extension to the existing membership is made.
addusermembership <type> <group> <user> <startDateSpec> <endDateSpec>	- Add a user to a user-based custom group. If overlap is detected, an extension to the existing membership is made.
backdatememberships <TIMESPEC> [type] [group]	- Backdate the start time of memberships that belong to the optionally specified type or type/group pair. If no type is passed, all memberships will be backdated. TIMESPEC may be specified as a number of seconds or as a time abbreviation such as 1m, 1h, 1d, or 1w.
addgroup <type> <group>	- Add a group of the specified type.
addgrouptype <membershipType><type>	- Add a group type based on the specified membership type (user or host).
renamegroup <old> <new>	- Rename a group.
renamegrouptype <old> <new>	- Rename a group type.
deletegroup <type> <group>	- Delete a group.
deletegrouptype <type>	- Delete a group type.
deletemember <type> <group> <member>	- Delete all memberships for the member and group specified.
deletemembership <type> <group> <member> <startDateSpec> <endDateSpec>	- Delete a specific membership for the member and group specified. Times must match an existing membership exactly.
dumpmemberships	- Dump existing memberships in config file format.
loadfromfile [external file path]	- Load membership information from configuration file. Default file is licmon.swd/config/accounts.cfg. Optionally, an external file may be specified.
loadfromoptions [external file path] <groupType> <optionsGroupList>	- Load membership information from FLEXlm options file. Specify a group type and the options file group(s) that will be used to populate a group by the same name. To use all groups, specify "all". A user cannot be a member of more than one group of the same type, so make sure different group types are specified for each group that contains a common user.
showgroups <type>	- Show custom groups of specified type.
showgrouptypes	- Show custom group types.
showmembers <type> <group>	- Show group members.
showmemberships	- Deprecated. Use showusermemberships.
showhostmemberships <host>	- Show group memberships for the specified host.
showusermemberships <user>	- Show group memberships for the specified user.
updatemembership <type> <group> <member> <oldStartDateSpec> <oldEndDateSpec> <newStartDateSpec> <newEndDateSpec>	- Update a membership. Times must match an existing membership exactly. This is normally used to shorten a membership due to how the addmembership action creates

extensions upon overlap.

ABOUT CUSTOM GROUP DATESPECS:

Times may be specified in the following formats:

10-digit Unix timestamp

YYYYMMDD

YYYY-MM-DD

Full date shown in "ftlm_accounts -showmemberships" output

Examples:

```
ftlm_accounts deletemembership location "San Jose" joe 1293655667 1293655668
```

```
ftlm_accounts deletemembership location "San Jose" joe 20100101 20110101
```

COMMON OPTIONS:

-f	- Force flag (use with reset to avoid dialog).
-v	- Increase verbosity.
-h	- This help.

EXAMPLES: DATABASE MANIPULATION:

```
% ftlm_accounts setdb "john mary dan" "ChipA"
```

```
% ftlm_accounts reset
```

```
% ftlm_accounts -f reset
```

```
% ftlm_accounts autofill
```

EXAMPLES: NEW CHECKOUTS:

```
% ftlm_accounts showlive
```

```
% ftlm_accounts setlive "john mary dan" "ChipA"
```

```
% ftlm_accounts setliveall
```

```
% ftlm_accounts clearlive
```

SEE ALSO:

ftlm_lmproject

Manually Set the Project for Live Checkouts

By using `ftlm_accounts` with either the `setlive` or `setliveall` actions, new checkouts can be assigned to project of your choosing.

The `setlive` action requires a user and project specification, whereas the `setliveall` action uses the configuration files to determine the project assignments. All live checkout assignments are preserved when the checkouts are loaded into the database.

Automatically Set the Project for Live Checkouts

For automatic updating of the project definitions for live checkouts, a periodic job is available. The **Configuration Information** page can be used to configure this task. Refer to the documentation for the [UPDATE_PROJECTS](#) task for more details.

Manually Annotate the Database with Project Information

It is possible to perform a post-processing on the database to modify the project for each checkout. This is required if the project configuration was not created during the initial Monitor installation.

Database post-processing is also handled by the `ftlm_accounts` utility. This process may take a while depending on the size of the database.

The `setdb` action sets the all checkouts in the database to the specified project for the specified user.

The `autofill` action automatically fills-in projects for all checkouts that have no project assigned. The information for the automatic filling is derived from three sources:

- The `licmon.swd/config/accounts.web.cfg` file.
- The `licmon.swd/config/accounts.cfg` file.
- If a user is not included in any of the projects defined in the configuration files, the utility then looks at the database information itself for previous assignments of projects to the user. If any are found, it picks the most frequently used project. Otherwise, a project of "none" is used.

Custom Group Configuration Activation

Manually manage custom groups

The `ftlm_accounts` utility is also used to manage custom groups. The utility can be used to show, create, update, and delete groups, group types, and group memberships. Memberships can be specified as arguments to the script or a map file can be used. See the usage syntax above for details on using these functions.

Custom groups can driven by LDAP. In this scenario, the `vovldap_update_memberships` utility is used to extract the data from LDAP, based on the [LDAP Integration](#), and populate it in the database. The utility can also be used to generate a map file instead, which can then be imported using the `ftlm_accounts` utility. The usage syntax for the `vovldap_update_memberships` utility is:

```
vovldap_update_memberships: Usage Message
```

DESCRIPTION:

```
This utility queries an LDAP server to look up configured user attributes
that are used to define group memberships in the LicenseMonitor
database. Resulting memberships begin at the time this command is
executed with durations of 10 years unless otherwise specified. If a user
has an active membership on one account and becomes a member of a
different account, the previous membership is terminated and the new one
begins at the time this command is executed.
```

USAGE:

```
% vovldap_update_memberships [OPTIONS]
```

OPTIONS:

```
-h                                -- This help.
-duration "timeSpec"             -- Specify the duration for the membership
                                  definitions.
                                  Time specification format is (Xd, Xw, Xm, Xy).
                                  Examples: 1d, 5d, 2w, 6m, 1y, 2y.
                                  Default duration is 1d.
-f "output_file"                 -- Save output to the <output_file> instead of
                                  writing results into the database. The @PROJ_DIR@
                                  variable can be used in the <output_file>
                                  specification which will be substituted with the
                                  full path to the server working directory (SWD).
-users "a b c"                   -- Use specified user list instead of obtaining user
                                  list from the database.
-v                                -- Increase verbosity.
```

EXAMPLES:

```
% vovldap_update_memberships
% vovldap_update_memberships -f
    /home/rtda/licmon/licmon.swd/config/accounts.ldap.cfg
% vovldap_update_memberships -f @PROJ_DIR@/config/accounts.ldap.cfg
```

Custom groups can also be driven by NIS. In this scenario, the `vovnis_update_memberships` utility is used to extract the data from NIS, based on the configuration defined in the accounts configuration file (see example above) and populate it in the database. The utility can also be used to generate a map file instead, which can then be imported using the `ftlm_accounts` utility. The usage syntax for the `vovnis_update_memberships` utility is:

`vovnis_update_memberships`: Usage Message

DESCRIPTION:

This utility queries NIS to look up group members that are used to define group memberships in the LicenseMonitor database. Resulting memberships begin at the time this command is executed with durations of 10 years unless otherwise specified. If a user has an active membership on one account and becomes a member of a different account, the previous membership is terminated and the new one begins at the time this command is executed.

USAGE:

```
% vovnis_update_memberships [OPTIONS]
```

OPTIONS:

```
-h -- This help.
-duration "timeSpec" -- Specify the duration for the membership
    definitions.
    Time specification format is (Xd, Xw, Xm, Xy).
    Examples: 1d, 5d, 2w, 6m, 1y, 2y.
    Default duration is 1d.
-f "output_file" -- Save output to the <output_file> instead of
    writing results into the database. The @PROJ_DIR@
    variable can be used in the <output_file>
    specification which will be substituted with the
    full path to the server working directory (SWD).
-users "a b c" -- Use specified user list instead of obtaining user
    list from the database.
-v -- Increase verbosity.
```

EXAMPLES:

```
% vovnis_update_memberships
% vovnis_update_memberships -f
    /home/rtda/licmon/licmon.swd/config/accounts.nis.cfg
% vovnis_update_memberships -f @PROJ_DIR@/config/accounts.nis.cfg
```

Automatically manage custom groups

For automatic updating of the custom group definitions from file, LDAP, or NIS, a periodic job is available. The Configuration Information page can be used to configure these tasks. Refer to the documentation for the [UPDATE_MEMBERSHIPS_*](#) tasks for more details.

Custom Groups Driven By FlexNet Publisher Options File

Custom groups can be obtained from an options file. This is mainly useful when there are user-locked licenses or reservations to groups defined in the options file. To extract and load these groups and their members into the Monitor database, use the `ftlm_accounts` utility (see above for usage details). It is important to make sure that the custom group type being specified for the group to extract be unique. For example, for the following user-locked license INCLUDE/GROUP pair in the options file:

```
GROUP nnu bob joe sue  
INCLUDE MATLAB:asset_info=12345 GROUP nnu
```

An appropriate call for the `ftlm_accounts` utility would be:

```
% ftlm_accounts loadfromoptions MLM.opt asset12345 nnu
```

This is required in case the `nnu` group is used for multiple user-locked licenses (asset tags). If there is only one usage of the `nnu` group in the options file, and there is no chance that it will be used with other asset tags in the future, then a more generic custom group type could be used, such as `MATLAB-NNU`.

Network Monitoring

Monitor can also be configured to monitor hardware and process information. These functions are enabled by launching a monitoring agent on each of the hosts to be monitored.

The agent connects to the Monitor server and periodically sends the following data:

- Machine load (UNIX only)
- CPU utilization
- Filesystem utilization
- Process listing
- Network traffic levels (Linux only)

Agents can be configured and controlled remotely from the Monitor server if certain conditions are met (see the Remotely Controlled Monitoring Agent Configuration section below). Otherwise, agents can also be configured locally on each machine to be monitored (see the Stand-alone Monitoring Agent Configuration section below).

Remotely Controlled Monitoring Agent Configuration

For remote control of agents, the following must be true:

- An Altair Accelerator installation must be available to the machine to be monitored (either network-based or local).
- An RSH, SSH, or `vovtssd` connection must be allowed between the Monitor server and the host to be monitored without a password prompt appearing. The `vovtssd` daemon is provided by Altair Accelerator and provides secure connection capabilities on any configurable port for the purpose of starting and stopping agents remotely. This daemon is especially useful for Windows, which does not provide built-in remote connectivity capabilities. See the Setting Up SSH Keys section below for

help with setting up SSH to not require a password, or the Setting Up `vovtsd` section below for help setting up the `vovtsd` daemon.

If the Altair Accelerator installation is not available on the remote machine, or none of the remote connection methods can be used, a stand-alone version of the agent must be launched directly on the remote machine. This type of agent cannot be controlled by the web interface. This is described in the Stand-Alone Monitor Agent Configuration section below.

Web-based Configuration

Visit the Agents administration page. This page can be used to add or delete agents on machines that possess remote connectivity capabilities (`rsh`/`ssh`/`vovtsd`):

Agents

An agent is required on every machine that is to be monitored for process and network information, or for performing remote license server administration tasks. If the Altair Software Package installation and remote connectivity (`ssh`, `rsh`, or `vovtsd`) to the remote machine are both available, use the form below to add agents. If the Altair Software Package installation or remote connectivity is not available, refer to the Administrator Guide for details on setting up agents locally on the remote machine.

#	Host	Type	Name	VOVDIR	Access Method	Status	Actions
	<input type="text"/>	monitor	<input type="text"/>	default	ssh		<button>Add Agent</button>
<div><div>Specify the remote host that will run the agent.</div><div>Use 'monitor' for monitoring-only agents, or 'manager' for agents that can also perform license management functions for ControlCenter (such as starting and stopping daemons).</div><div>The name is normally the concatenation of type and host name, but can be anything as long as it is unique.</div><div>If the installation directory (VOVDIR) is different between the Altair Monitor server and the agent machine, specify the VOVDIR location as seen by the agent here. Use 'default' if the paths are equivalent or if using <code>vovtsd</code> to control the agent. Use <code>/</code> in paths, even on Windows. Examples are <code>/opt/altair/2019.01/linux64</code> and <code>c:/altair/2019.01/win64</code>.</div><div>Method to access target host. If remote, the method must work without a password as denby from <code>uslt70</code>. For UNIX, consider either <code>ssh</code> or <code>rsh</code>. For Windows, either OpenSSH must be installed or Altair's <code>vovtsd</code> communication daemon must be used. The <code>vovtsd</code> port is tested no more than once every 5 minutes.</div></div> <div><div>Start the configured agents that are not started yet. It may take some time. Please click on the "Agents" link in the menu again to reload after a while.</div><div><button>start all</button></div></div>							

For the advanced administrator that needs batch configuration of the monitoring agents, here are the links to the files to edit:

- [c:/altair/swd/licmon.swd/taskers.tcl](#)
This file is described [here](#).

Figure 21: Agents Options

Add agents using the web form on this page. Specify the host, the type of agent ("lm" for monitoring agents, "licmgr" for license management agents, which are described later in this administrator guide), and adjust the agent name to be unique, if needed. This will be the case if more than one agent will be running on the same remote machine. This is normally only true for license management agents. Finally, specify the access method (`rsh`/`ssh`/`vovtsd`) and click **Add Monitoring Agent** to register the agent with the system. The agent can now be started by clicking **start all agents**.

Note that for agents started via `rsh` or `ssh`, the value of the Altair Accelerator installation directory (`$VOVDIR`) for the agent defaults to the value of `$VOVDIR` for the Monitor instance. For `vovtsd` agents, the value defaults to the value of `$VOVDIR` for the `vovtsd` instance. If a different value of `$VOVDIR` is desired, an override can be specified by using the file-based configuration method below.

File-based Configuration

To configure monitoring agents, edit the `licmon.swd/tasker.table.tcl` file and add the hosts that are to be monitored. For example:

```
DefineTaskerWithArgs HOSTNAME TYPE -rshcmd REMOTECMD -name AGENTNAME
```

- The value of `HOSTNAME` should be the DNS name or IP address of the host to be monitored.
- For monitoring agents, the `TYPE` should be "lm". For license management agents, the `TYPE` should be "licmgr".
- The `REMOTECMD` value should be one of `rsh`, `ssh`, or `vovtsd`, depending on the connection method desired. If using `vovtsd`, the argument "`-vovtsdport PORT`" arguments must be passed as well, where `PORT` is the port number on which `vovtsd` is running on the remote machine. For example:

```
DefineWithArgs dragon lm -rshcmd vovtsd -name lmdragon -vovtsdport 16666
```

If the Altair Accelerator installation directory location (`$VOVDIR`) needs to be overridden use the `-vovdir` option to specify the desired location:

```
DefineWithArgs dragon lm -rshcmd vovtsd -name lmdragon -vovtsdport 16666 \  
-vovdir /opt/rtda/2023.1.0/linux
```

Once the agent has been properly configured, it can now be started by using the **start all agents** button on the page mentioned above, or by issuing the following commands at the CLI:

```
% vovproject enable licmon  
% start
```

By default, this will start all agents that are not already started. To start an individual agent, pass the agent's name into the `vovtaskermgr start` command.

Setting Up SSH Keys (UNIX only)

For networks where home directories are stored remotely and available to all machines via NFS, Monitor provides a utility called `vovsshsetup` to setup SSH keys, which allow SSH to connect to a remote machine without prompting for a password. After running the utility, one manual connection will need to be performed to each of the hosts so that its host key is added to the `known_hosts` file. Each host also must be able to mount the filesystem containing the Monitor software installation.

Setting Up vovtsd

UNIX

To manually start `vovtsd`, execute the daemon in a shell:

```
% source /<install_path>/<version>/common/etc/vovrc.csh (or vovrc.sh for bash)  
% vovtsd -port 16666
```

The complete usage syntax of `vovtsd` is:

```
vovtsd: Usage Message
```

```
VOVTSd: Vov Tasker Service Daemon  
This utility listens for requests to launch taskers for
```



```
various projects, but always for the same user.  
The requests typically come from vovtaskermgr.
```

USAGE:

```
% vovtsd [OPTIONS]
```

OPTIONS:

```
-v                -- Increase verbosity.  
-h                -- Print this help.  
-debug           -- Generate verbose output.  
-help            -- This message.  
-normal          -- Start a normal daemon (for current user)  
-expire <TIMESPEC> -- Exit from vovtsd after specified time.  
-user <user>     -- Specify the user that should be impersonated.  
                  vovtsd computes the port number by  
                  hashing the user name.  
-port <n>        -- Specify port to listen to.  
-requireauth <n> -- Require key-based auth from clients.  
                  Client public keys need to be set in  
                  VOVTSD_USERKEYS env var if enabled.  
-userkeys "<key1> <key2> ..." -- Specify a list of public keys that are  
                  allowed to authenticate the client.
```

EXAMPLES:

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

To enable automatically starting `vovtsd` upon UNIX system startup, use the available example `.bat` files as a guide to create your own script, and place it in the appropriate directory. Example startup files are provided in `$VOVDIR/etc/boot`. Choose the one that best fits your scenario.

Windows

First, download the `vovtsd-win32.exe` or `vovtsd-win64.exe` program from the release distribution area of the Altair website. To run `vovtsd` manually, simply double click the program and fill out the form:

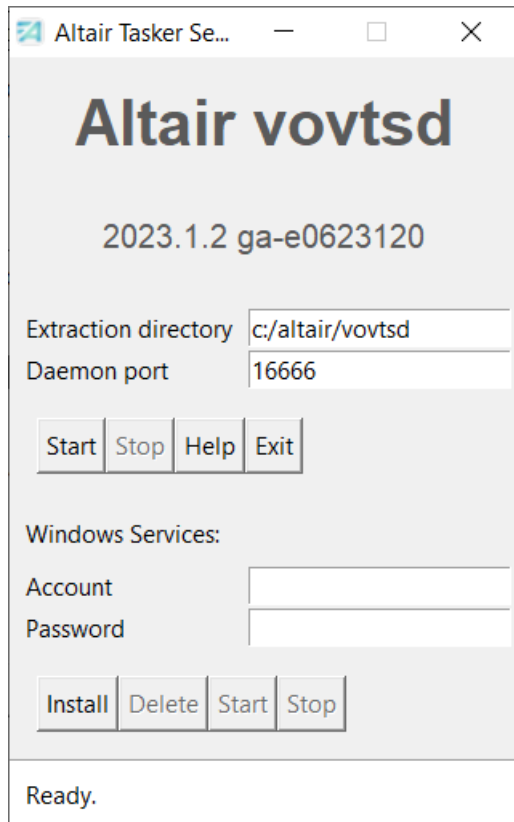



Figure 22:

To have `vovtsd` run automatically via a Windows service, launch the program with administrator privileges by right-clicking and selecting the **Run as administrator** option and use the Windows Services area to configure the `vovtsd` service.

 **Note:** If upgrading from a version less than 2015.03, first stop and uninstall the old service before installing the new service.

Stand-alone Monitoring Agent Configuration

If network security measures prevent any of the above methods, a stand-alone version of the agent can be installed and configured locally on each of the remote machines to be monitored. This agent is designed to reconnect to the Monitor server in the event the connection is interrupted.

UNIX

First, download the `vovtsd-VOVARCH` program from the release distribution area of the Altair website, where VOVARCH matches the desired architecture (eg `linux64`). To run the agent manually, simply double click the program and fill out the form for the "Monitor" agent type ("Manager" agents are covered in a later chapter):

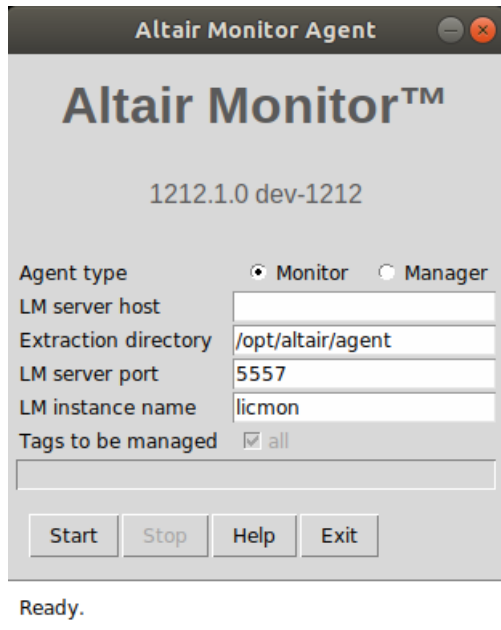


Figure 23:

The agent can be executed with the `-h` option to get usage syntax help, which can be used to develop a boot script for starting the agent automatically.

Windows

First, download the `lmagent-win64.exe` program from the release distribution area of the Altair website. To run the agent manually, simply double click the program and fill out the form for the "Monitor" agent type ("Manager" agents are covered in a later chapter):

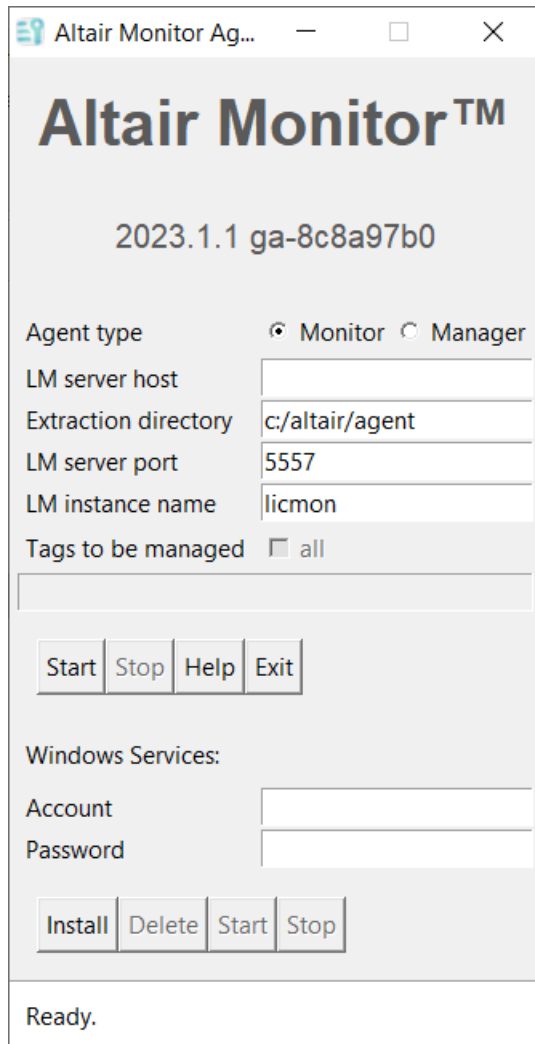


Figure 24:

If not using an legacy license key file, make sure to specify the location of the ALM license server or file.

To have `lmagent` run automatically via a Windows service, launch the program with administrator privileges by right-clicking and selecting the **Run as administrator** option and use the Windows Services area to configure the `vovtsd` service.

Monitor Processes

For tools that are not licensed, usage tracking can still be achieved.

Monitor supports two methods for doing this: process tracking and [tool wrapping](#).

The process tracking method works by tracking the existence of the processes of the tool invocations, with the following limitations and assumptions:

- Only available for tools that run on machines that have [monitoring agents](#) configured and running.

- A tool invocation must have one unique executable name that shows up in the process list, and one such process represents one and only one "license" usage.

Web-based Configuration

The form in the Monitor administration web page contains a monitor type for process monitoring.

File-based Configuration

Once process monitoring agents are configured, edit the `licmon.swd/vovlmd/config.tcl` file to setup the processes that are to be monitored using `add_PROCESS_TRACKING`.

Wrapping Unlicensed Tools

The wrapping technique for monitoring unlicensed tools uses Monitor's built-in license management capability. This basic capability is activated by first creating a served license, then instructing the tool to either count its run as a checkout, or to request a license before being checked out, via a script or binary wrapper. Monitoring occurs automatically.

Creating a License

The `vovlmd` daemon configuration file, found at `licmon.swd/vovlmd/config.tcl`, allows for the creation of served licenses. The following is an excerpt from the `vovlmd` configuration template which details how to create a served license.

```
#####
### SERVED LICENSES ###
#####

# The following line is required to register licenses to be served.
# Uncomment it to activate served licenses.
# vtkle_features_init

# Create a license
# vtkle_feature_set <NAME> <CAPACITY|unlimited> [OPTIONS]

### Options
#
# -expire TIMESPEC|never  -- Set the expiration time for the license. Can be
#                           specified in seconds-from-now or a VOV TIMESPEC
#                           (1d, 1w, 1y). Default is to expire in one hour.
#                           Note: 1m is 1 minute, not 1 month!
# -log                     -- Generate a log of activity for debugging purposes.
#                           The log is located at licmon.swd/logs/features.log.

### Examples
#
# vtkle_feature_set sqlite 3 -expire never -log
# vtkle_feature_set sqlite unlimited -expire 180d
```

Using a License

Using the capability requires an environment variable that points to the location of the licenses being served. This variable is in the format of:

```
% setenv VOV_JOBCOUNTER 5555@lmsrv,licmon
```

In the example above, `lmsrv` is the Monitor server name, `5555` is the port (which is the default), and `licmon` is the Monitor instance name (also default).

The license then must be requested by using the `vtool` wrapper:

```
% vtool LICENSENAME LICENSEQTY TOOL
```

The wrapper requires 2 arguments (the license feature requested and the number of tokens required), which are followed by the command to be executed, along with the command's arguments. The complete usage syntax for the `vtool` wrapper is:

`vtool`: Usage Message

DESCRIPTION:

A tool to execute a command using one or more features managed by the Altair Engineering-LM license manager. The `VOV_JOBCOUNTER` environment variable must point to the license manager `vovserver` instance using the format of `port@host,instance` (eg: `5557@lmhost,licmon`).

Multiple instances can be specified by separating them with a colon (eg: `5557@lmhost1,licmon1:6667@lmhost2,licmon2`). If multiple instances are specified, each subsequent instance will be queried if the requested tokens were not granted by the previous instance in the list. By default, each instance must be able to grant the requested number of tokens in whole. However, scavenging mode (`-s`) can be used to allow each instance to partially fulfill the request.

USAGE:

```
% vtool [-v] [-refresh <S>] [-jobid <I>] -f <feature1> <tokens1> \  
[-f <feature2> <tokens2> ...] <command ...>
```

OPTIONS:

<code>-v</code>	-- Increase verbosity.
<code>-f <feature> <N></code>	-- Check-out N tokens of specified feature. If N is negative, queueing will be enabled for the absolute value of N tokens. It is an error for N to be zero.
<code>-refresh <S></code>	-- Specify refresh cycle (time-spec) for tokens (default is 600 seconds), meaning that the tokens are refreshed every 10 minutes. Setting a refresh cycle of 0 disables the refresh.
<code>-jobid <I></code>	-- Specify a jobid. If a refresh is performed and the jobid status is Failed, then the command is terminated with <code>vovkill</code> .
<code>-s</code>	-- Scavenging mode. When <code>VOV_JOBCOUNTER</code> is set to a colon delimited list of license servers, a check out from each server will occur until the target number of tokens has been reached. If queueing is enabled, only the last server will enter queueing mode regardless of its license capability.

```
SIGNALS:
  TSTP      -- Release licenses and suspend execution.
  CONT      -- Resume execution.
  HUP       -- Caught and ignored
  TERM      -- Caught and ignored
  QUIT      -- Caught and ignored
```

```
SEE ALSO:
  vlmstat vovcounters
```

```
EXAMPLES:
% vtool -f lic_drc -1 -f lic_hdrc -1 sleep 10
% vtool -s -f lic_drc 3 sleep 10
% vtool -f lic_drc 3 -refresh 2m sleep 10000
```

Script Wrapping Example

Suppose there is a tool called `spice` which performs circuit simulation and typically runs in a few seconds. The tool uses the feature `expensive_spice`. Because the tool is so fast, the sampling methodology will capture only a small fraction of the check-outs of `expensive_spice`. The tool's invocation command is: `% spice netlist.spi`.

- Set the environment variable `VOV_JOBCOUNTER` to point at the Monitor server:

```
% setenv VOV_JOBCOUNTER 5555@HOSTNAME,licmon
```

- Create a new license called `mirror_spice`. In the `vovlmd/config.tcl` file add:

```
% vtkle_features_init
% vtk_feature_set mirror_spice 10 -expire never -log
```

- Test the license availability:

```
% vovcounters
vovcounters Wed 17:14:15: message: JobCounter: 5555@bear,licmon
vovcounters Wed 17:14:15: message: Wed Oct 10 05:14:15 PM PDT 2007
Counter spice                      0 of 5
```

Another utility, called `vlmstat` has been provided that provides a look and feel similar to `lmstat`.

- Create an instrumented script, also called `spice` as follows:

```
#!/bin/csh -f
# This is the 'instrumented spice'
vtool mirror_spice 1 /full/path/to/spice $1
```

- Using the `PATH` search mechanism, make sure that when the users invoke `spice` they find the instrumented script before they find the real `spice`.

The overhead introduced by the instrumented script is negligible. The statistics collected for the counter `mirror_spice` will be fairly accurate, but not necessarily an exact, representation of the utilization of `expensive_license`, in those cases in which the job actually checks out the license many seconds after it starts or checks it in many seconds before it exits.

Binary Wrapping

A binary wrapper can also be created for a tool, instead of using `vtool`. A utility is provided that helps in the creation of the binary wrapper: `vovprotectwithcounter.csh`.

Description:

A utility to compile a binary wrapper for a binary in order to be able to count all invocations of that binary.

The utility compiles a small C program linked to the `libcounters.a` library.

This utility is meant to be an example of how you can use the `libcounters.a` library.

You are free to look at the details of the script (`$VOVDIR/scripts/vovprotectwithcounter.csh`) and modify it to suit your particular needs.

To change the C compiler, set the variable `CC`.

The output is a binary in the current directory with the same name as the wrapped binary.

Usage:

```
% vovprotectwithcounter.csh [-debug] /full/path/to/binary countername tokens
```

Example:

```
% vovprotectwithcounter.csh /bin/cp count_cp 1
% vovprotectwithcounter.csh -debug /bin/cp count_cp 1
% env CC="gcc -m64 -g" vovprotectwithcounter.csh /bin/cp count_cp 1

set B = "`basename $0`:"
set B = `basename /integ/zfbuild/integ/builds/201901release/build/common/scripts/
vovprotectwithcounter.csh`:
basename /integ/zfbuild/integ/builds/201901release/build/common/scripts/
vovprotectwithcounter.csh

if ( $#argv < 3 ) then
if ( 0 < 3 ) then
cat << EOF
cat
exit 2
exit 2
```


License Server Management

License Management Overview

The Monitor ControlCenter performs license server management tasks such as:

- Starting and stopping license daemons
- Rereading license files
- Version tracking of licenses and options files
- License files can be fetched from a license server, edited within the web interface, then published back to the license server
- New license files can be uploaded and distributed to remote license manager directories
- New license server directories can be created from scratch

ControlCenter requires an agent program to run on each license server host to be managed. The agent is responsible for running supported administration commands, such as starting and stopping a license daemon, as well as transmitting files to and from the server.

Server-side storage and configuration management of the files is handled via the lightweight CVS revision control system. For UNIX platforms, please make sure the CVS program is installed and located in the path variable in the Monitor environment. For Windows, CVS is bundled with Monitor.

The ControlCenter functionality is accessible via the ControlCenter link in the web interface.

Get Started with ControlCenter

1. To begin using ControlCenter, click on its tab in the Monitor web interface.
This brings up the ControlCenter cockpit. The cockpit is where high-level license server management can be performed. This includes starting, stopping, and issuing license reread requests to license servers. Upon first access, the cockpit will be empty.
2. To begin ControlCenter setup, click the **Setup** menu option, then click **Initialize Repository**.
This creates and initializes the CVS repository that is used to store multiple versions of licensing files.

License Management Remote Agents
List of the running remote agents, which are required for management tasks. Click [here](#) for more details on the connected agents.

Name	Host	Owner	Status	Managed Tags
<div> ? <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> </div>				

Showing 0 out of 0 rows | Limit rows to display: 10 | [Filter](#) ☐ ignore case

[Manage Agents](#)

Agents are [vovtasks](#) that provide the following resources: Imagent OWNER:* TAG:*
The page that shows all taskers can be reached [here](#).

Figure 25: ControlCenter setup before CVS initialization

A box will appear at the top of the page that contains the output of the initialization.

3. If unsuccessful, check the output for any corrective actions and try again. If successful, a green check will replace the button:

Server Overview Server & File Control Agent Status Setup

```

project-id: d85d9ac293cac126637601484383c25d9d1830da
server-id: 3e01959dbb32d48e4ad60d5b26941f893e5d534e
admin-user: denby (initial password is "Ik8to3aMUx")
project-name:
repository: c:\altair\swd\licmon.swd\licmgr\repo.fossil
local-root: c:\altair\swd\licmon.swd\licmgr\current\root/
config-db: C:/Users/denby/AppData/Local/_fossil
project-code: d85d9ac293cac126637601484383c25d9d1830da
checkout: d89bcf92e664fd8ac6f472ab175b2e6b3d1f4ef8 2023-08-10 17:47:46 UTC
tags: trunk
comment: initial empty check-in (user: denby)
check-ins: 1

Done.
  
```

Configure License Server Control
Below are the steps needed to setup the ControlCenter for use. Come back to this page when new license servers are added or to create a new license server from scratch.

Initialize the change tracking system:

✓ Repository has been initialized

Figure 26: ControlCenter setup after CVS initialization

4. Each daemon that is to be managed by ControlCenter will need to be configured. To get started with the configuration, move to the **Create Configuration File Templates** section and choose the appropriate scenario.

The first scenario is used to create templates for all existing monitors. The second scenario is used when create a new license server that is not yet being monitored. Again, the output from this step will be shown at the top of the page:

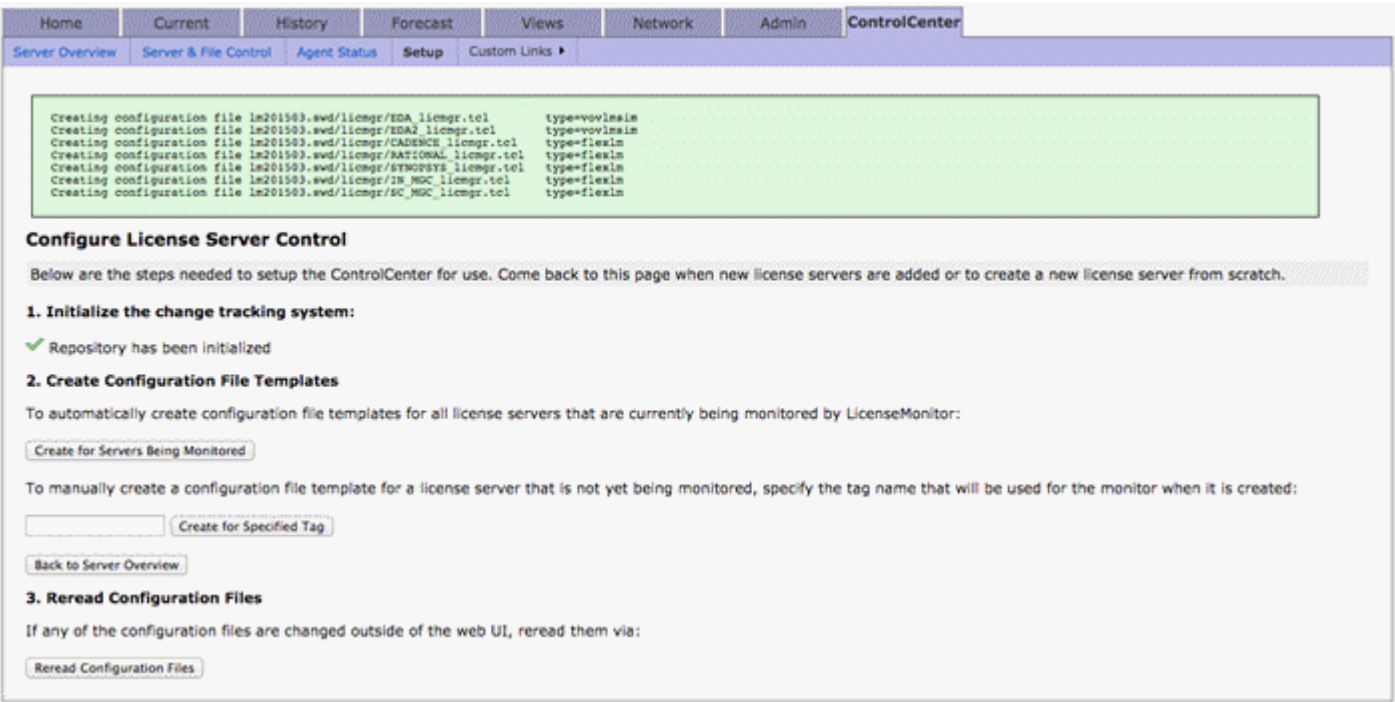


Figure 27: ControlCenter setup after creation of the templates

5. Once the templates have been created, click **Back to Server Overview** to be taken back to the cockpit. From here, a tag can be chosen for further configuration:

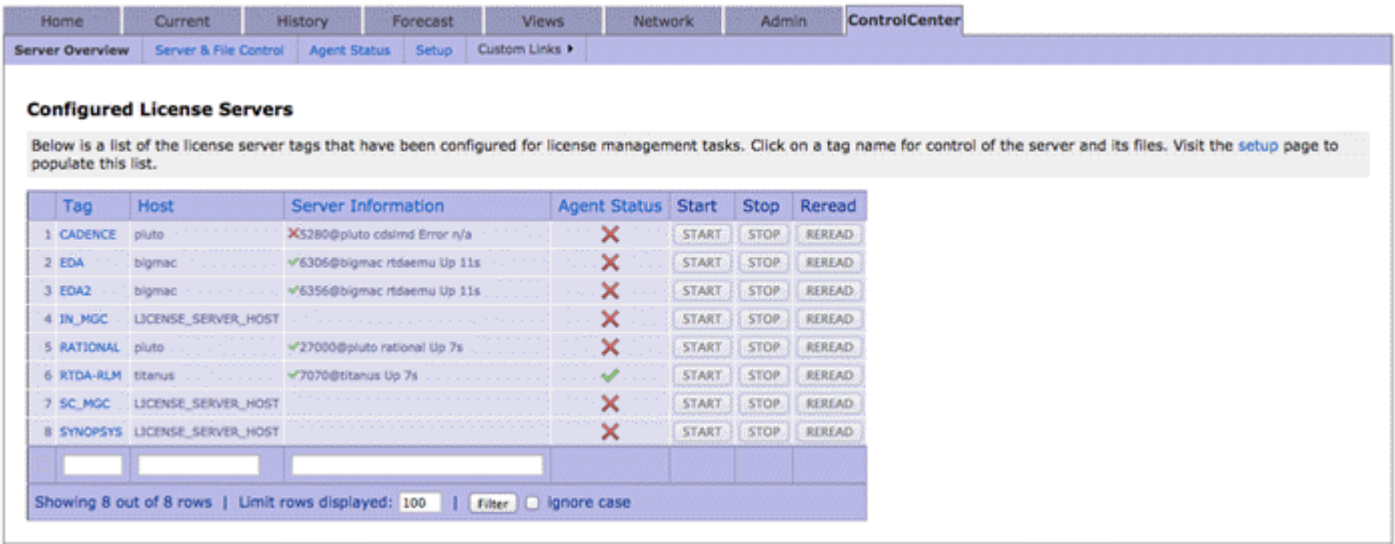


Figure 28: ControlCenter cockpit

6. Select a tag by clicking on its name in the table.
In the example below, the "RTDA-RLM" tag was clicked. This brings up the **Server & File Control** page, which shows the information for the license server instance that the tag represents:

Home **Current** **History** **Forecast** **Views** **Network** **Admin** **ControlCenter**

Server Overview **Server & File Control** **Agent Status** **Setup** **Custom Links**

License Files and Commands

Configure license management, work with licensing files, and control license servers. Initially, only the LicenseMonitor owner-user has permission to add additional managers.

License server tag:

Agent status:

Below is the status of the agent(s) that are managing this license server. FLEXlm triad instances must have an agent on each of the three machines for complete functionality.

✓ titanus

Configuration:

```
### Configuration for reprise tag RTDA-RLM
### Use forward slashes in all paths
LICMGR_JOB HOSTS RTDA-RLM "titanus"
LICMGR_JOB DIR RTDA-RLM "PATH_TO_LICENSE_DIRECTORY"
LICMGR_JOB START RTDA-RLM "sh -c LICENSE_FILE_NAME -dows >& RTDA-RLM.HOST#.log &"
LICMGR_JOB REREAD RTDA-RLM "reread -c 7070@titanus"
LICMGR_JOB STOP RTDA-RLM "rmdown -c 7070@titanus &N -q"
LICMGR_JOB FILES RTDA-RLM "LICENSE_FILE_NAME"
LICMGR_JOB LOGDIR RTDA-RLM ""
LICMGR_JOB OWNER RTDA-RLM ""
LICMGR_JOB MANAGERS RTDA-RLM {}
LICMGR_JOB COMMENT RTDA-RLM ""
LICMGR_JOB ENABLED RTDA-RLM 0; # SET TO 1 TO ENABLE
```

License server control:

Please enable and configure license management for this tag in the configuration area above.

License file control:

Please enable and configure license management for this tag in the configuration area above.

Figure 29: Tag Details page showing the RTDA-RLM tag

The configuration template for the RTDA-RLM tag that was generated in the above step, `licmon.swd/licmgr/RTDA-RLM_licmgr.tcl`, is shown. The configuration can be edited and saved from within this page. The configuration utilizes the LICMGR_JOB procedure, which is used to register parameters for the given license server. The parameters include the following:

- **HOSTS:** show the host on which the license server instance is running. This is normally one or three hosts. Multiple hosts should be separated by a space.
- **DIR:** the directory in which the license server instance is to be started. This is normally a single directory, but could be a list of directories if HOSTS refers to a triplet and the directory is different on each of the different hosts. If the HOSTS line shows more than one host, then the first directory is used for the first host, the second for the second host and so on. If there are less directories than hosts, then the last directory is used for each host in excess to the number of directories.
- **LOGDIR:** The default location for license manager job logs is the directory specified by the DIR setting described above. If another location is desired for these logs, specify the absolute path with the LOGDIR setting.
- **START:** the command to start the daemon. This command will be executed in the directory specified above. This command can be empty if the start capability is not wanted.
- **STOP:** the command to stop the daemon; this command will also be executed in the directory specified above. This command can be empty if the stop capability is not wanted.
- **REREAD:** the command to force a reread of the license file. This command can be empty if the reread capability is not wanted.

- **FILES:** this is a list of paths relative to the directory DIR. Normally this is the license file and the options file, but the list may also include other files. These are the files that can be fetched and deployed by LicManager. The list can be empty if file fetching and publication capabilities are not wanted.
 - **OWNER:** the user that owns the license daemon. Ideally, this should be the same user as the owner of Altair Monitor, but in many cases this must be a different user. This user exists in the remote system(s) specified in the HOSTS line. If left empty, it is assumed that the owner is the same as the owner of Altair Monitor.
 - **MANAGERS:** the users of Altair Monitor that have authorization to perform license management tasks. If left empty, it is assumed that the only manager is the same as the owner of Altair Monitor.
 - **COMMENT:** a general-purpose comment field that can be used to communicate status between team members.
 - **ENABLED:** a flag to enable the processing of the configuration file. To enable the configuration, set the flag to 1.
- 7.** ControlCenter will try to fill in reasonable defaults for those tags that are already being monitored by using Altair Monitor's tracked data. Once the configuration is appropriate for the tag being setup, click the "Save" button, which saves the configuration and creates the jobs for starting, stopping and rereading the daemon, as well as the fetch and publish jobs for each managed file specified in the configuration.
- 8.** The LicManager functionality, such as starting and stopping the daemon as well as fetching and deploying of files, depends on the availability of "LicManager Agents" running on the remote hosts. Agents are discussed in the next section. Once agents are configured and running, the cockpit can be used to start/stop/reread, and the Server & File Control page can be used do the same, as well as to perform file transfer to and from the license server:

License file control:

Click the edit button to modify a licensing file, then return to this page to deploy it. to the license server. Once a file is deployed, it can be read into the license manager via the reread button above. Note that not all file changes can be reread and may require a full restart of the license manager instead.

Unknown file type 'rtde.lic':

Local copy of file not found. Import a new file or fetch the existing one from the server.

Figure 30: File transfer

- 9.** Each configured licensing file can be fetched from the remote host by clicking on the "Fetch" button. The file is staged from the remote host into the CVS repository.
- 10.** If a new license file has been received from a vendor, ControlCenter can be used to import the file (click "Import") by either uploading the file or pasting it into a form.

The screenshot displays the 'ControlCenter' tab in the Altair Monitor application. The top navigation bar includes tabs for Home, Current, History, Forecast, Views, Network, Admin, and ControlCenter. Below this, a sub-navigation bar shows 'Server Overview', 'Server & File Control' (which is active), 'Agent Status', 'Setup', and 'Custom Links'. The main content area is titled 'Import File \"/>

Figure 31: File import

11. Click **Deploy** to stage the licensing file to all configured license server hosts.
12. Once a file has been fetched or imported, it can be quickly viewed by clicking **View**, or it can be edited via the **Edit** button. From the edit screen, multiple versions can be managed. When a change is made and saved, the changes can be checked into the CVS repository. By default, the latest version (the "head") is the active version of the file used for deployment. However, any historical version can be checked out to be the active version, and that version would be deployed instead.



Figure 32: File edit

13. After deployment of licensing files, the license manager will need to be instructed to reread the license files. This can be performed by clicking on the "reread" command for each of the hosts to execute the configured reread function.

Agent Configuration

Agent List

A license management agent is a program that runs on a remote host that accepts and executes commands that are issued from the Monitor server, as well as performs file transfers to and from the remote host. Agents require either network access to the Altair Accelerator product installation or a local installation on the remote host.

ControlCenter license management agents are viewed using the **Agents** page. Initially, the list of agents will be empty.

License Management Remote Agents
List of the running remote agents, which are required for management tasks. Click [here](#) for more details on the connected agents.

Name	Host	Owner	Status	Managed Tags
<div> ? <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> </div>				
Showing 0 out of 0 rows Limit rows to display: <input type="text" value="10"/> Filter <input type="checkbox"/> ignore case				

[Manage Agents](#)

Agents are [vovtasks](#) that provide the following resources: Imagent OWNER:* TAG:*
The page that shows all taskers can be reached [here](#).

Figure 33: ControlCenter License Management Agents

Click **Manage Agents** to bring up the Agent Management page for agents that can be remotely controlled via ssh, rsh, or vovtsd. ControlCenter's license management agents are configured in the same manner as monitoring agents with the exception that each agent can be assigned to a specific tag(s) so that it will only interact with the license server(s) to which the tag(s) are related. This allows for securing tags to their respective administrator personnel. If such security is not required, the agent can be configured to interact with all tags found on the same machine on which the agent is running. Refer to the [Network Monitoring](#) for setting up license management agents as well.

Command Line Setup

An advanced user may want to use the command line to setup LicManager. All necessary files are in the directory `licmon.swd/licmgr` and the utility to use is called `ftlm_licmgr_init`.

The utility `ftlm_licmgr_init` can be used to initialize the license management subsystem from the command line. Here is its usage:

```
ftlm_licmgr_init: Usage Message

DESCRIPTION:
  Initialize the licmgr/current directory for Monitor.
  This is the directory where copies of the active licenses
  are stored.
  This script is normally called automatically by ftlm_licmgr.cgi.

OPTIONS:
  -tag TAG      -- Only update for specified tag
  -cm           -- Initialize the code management system
  -templates    -- Initialize the templates for each known tag
  -reread       -- Reread configuration files.
  -v           -- Increase verbosity
  -h           -- This help message

EXAMPLES:
  % ftlm_licmgr_init -cm
```



```
% ftlm_licmgr_init -templates  
% ftlm_licmgr_init -templates -tag MGC
```

The common sequence to initialize the system is the following:

- Initialize CVS.

```
% ftlm_licmgr_init -initcvs
```

This step may fail if CVS is not available on your Monitor server machine.

- Create the initial templates.

```
% ftlm_licmgr_init -templates
```

- Edit the templates to activate them. This is mostly done manually, since a lot of the information required to start and stop the daemons is not automatically retrievable from `lmstat`.

```
% ftlm_licmgr_init -f
```

- Create the jobs in the system. This step needs to be repeated each time the configuration files are changed.

Other Configurations

Notification

Monitor includes a facility for e-mailing notifications of events that require attention, such as notifications of license daemons that are down or licenses that are checked out for long periods of time. The `vovnotifyd` daemon is responsible for sending out notifications.

Web-based Configuration

Using the browser interface, visit the **Notification** administration page. If visiting this page for the first time, click on the **Activate** link to enable notification. This step creates the `licmon.swd/vovnotifyd` directory and the required configuration file, then displays the health check information page:

Notification Configuration
Use the forms below to configure the notification system.

Procedure	Status	Frequency	Check Mail	Recipients	Actions
CheckAlerts	✓	10m00s	1d00h	@OWNER@	Edit Disable
CheckDownLicDaemon	✓	10m00s	1d00h	@OWNER@	Edit Disable
CheckDownTasks	✓	10m00s	1d00h	@OWNER@	Edit Disable
CheckVendorLicenseExpiration	✓	10m00s	1d00h	@OWNER@	Edit Disable
Daemons	✓	10m00s	1d00h	@OWNER@	Edit Disable
FailoverServerCandidates	✓	10m00s	1d00h	@OWNER@	Edit Disable
LicenseNearSaturation	✓	10m00s	1d00h	@OWNER@	Edit Disable
LongCheckouts	✓	10m00s	1d00h	@OWNER@	Edit Disable
ServerDiskSpace	✓	10m00s	1d00h	@OWNER@	Edit Disable
ServerSize	✓	10m00s	1d00h	@OWNER@	Edit Disable

Figure 34: Admin Tab - Notification

This page includes a list of the various health checks that will be performed which can trigger notification e-mails. The actions column allow for enabling/disabling, or editing these checks. Each health check has certain configurable parameters that adjust things such as check and mail frequency, as well as check-specific settings. This page manages the `licmon.swd/vovnotifyd/config_aux.tcl` file.

Before configuring the health checks, the STMP configuration needs to be established so that email can be sent from the Monitor notification system. This is done by visiting the SMTP configuration page, available from the notification drop-down menu:

Notification Configuration

Use the forms below to configure the notification system.

SMTP Server	localhost
The name of a host that is running SMTP.	
SMTP Port	25
Normally 25.	
SMTP Sender	Altair Engineering-notification
The sender of the message. Some SMTP agents require that this is a valid email address.	
Recipient Domain	
Used to compute the recipient email address from the user name.	
Project Host Domain	
Domain used in URLs found in notification emails.	
Home URL	
URL used to get back to this vovserver. This can be set using property NOTIFY_HOMEURL on object 1. Leave blank for default.	
Timeout (Time Specification)	2m00s
The time the daemon waits for events from the server. After each timeout, the daemon performs the health checks. Typical value is '2m'.	
Administrators	denby
Specify one or more space separated e-mail addresses.	

Figure 35: SMTP Configuration Page

Fill out the server settings form with the values that will enable a connection with a local STMP server, save, then send a test mail to confirm that the configuration is correct and that mail can be sent.

In some environments, the user name that is used to login to Monitor is not used as the user's email address. For these situations, fill in the **User email map** page with user name to email address mappings. This page is also available on the notification drop-down menu:

Notification Configuration

Use the forms below to configure the notification system.

Emails are sent directly to the appropriate user IDs (or *user@sourcedomain*) by default. Use this form to define alternative email addresses, so that the email for a user will be sent to the configured corresponding email address.

User Id	Email Address	Action
rtдамгр	john@mydomain.com	<button>Delete</button>
<input type="text"/>	<input type="text"/>	<button>Add</button>

Figure 36: Email Map Page

File-based Configuration

The main configuration file for `vovnotifyd` is the `licmon.swd/vovnotifyd/config.tcl` file. A complete, documented example file exists in the installation tree in the `common/etc/config/vovnotifyd` subdirectory of the installation root:

```
# Notification configuration file.
# Should be placed in the vovnotifyd directory of the .swd.
# All settings are required unless specified otherwise.
# Unused optional settings should be commented out.

# Create an e-mail address map, stackable, optional
addUserToEmailAddressMap  rtдамгр john@mydomain.com

### Monitor-specific settings
# See notification configuration documentation in Monitor Admin Guide
# ConfigureTag      TAG OPTION VALUE
# ConfigureFeature   FEATURE OPTION VALUE

### Examples:
# ConfigureTag      MGC -poc { john mary }
# ConfigureFeature   EDA/MATLAB -longcheckout 2d -userlongcheckout john 1w -mincap 5
#                   -triggerperc 90
# ConfigureFeature   SIMULINK -poc bob -mincap 10 -triggeruse 12
```

These parameters should be modified to match that of the SMTP server environment. The `vovnotifyd` daemon will need to be restarted after making changes to this file. For more information on daemon control, visit the [Daemon](#) topic.

Debugging

If the notification configuration does not appear to work, `vovnotifyd` can be launched in the foreground with increased output verbosity, to aid in debugging the configuration. To do this, first shut down the daemon, then:

```
% vovproject enable licmon
% cd `vovserverdir -p vovnotifyd`
% vovnotifyd -v -v
```

Feature and Tag Specific Configurations

Some health checks allow for tag or feature specific configurations. For example, the check for long checkouts (`doTestHealthLongCheckouts`) works best when you define the meaning of "long" on a per-feature basis. The notification configuration web page, as well as the list below, details the available options.

To configure the health checks, edit the `licmon.swd/vovnotifyd/config.tcl` file and configure the desired features and/or tags with either `ConfigureTag` or `ConfigureFeature`. When redundant settings are found between a tag and a feature, the most specific one wins. In `ConfigureFeature`, features may be specified in `FEATURE` form or `TAG/FEATURE` form. Available options are as follows:

ConfigureTag TAG -poc { USER1 USER2 USERN }

Specify the POC list for the specified tag. These users will be notified for feature-related notifications in addition to the LM administrator user(s).

ConfigureTag TAG - longcheckout TIMESPEC

Specifies the tag-wide long checkout trigger threshold. Set to 0 to disable.

ConfigureTag TAG - userlongcheckout USER TIMESPEC

Specifies a tag and user-specific long checkout trigger threshold for the specified feature. Set to 0 to disable.

ConfigureTag TAG -mincap INTEGER

Specifies a tag-wide minimum capacity for features to be considered for the saturation notification.

ConfigureTag TAG -triggerperc INTEGER

Specifies a tag-wide saturation trigger percentage threshold.

ConfigureTag TAG - daystoexpire INTEGER

Specifies the expiration trigger threshold, in days, for the specified tag.

ConfigureFeature FEATURE - poc { USER1 USER2 USERN }

Specify the POC list for the specified feature. These users will be notified for tag-related notifications in addition to the Altair Monitor administrator user(s).

ConfigureFeature FEATURE - longcheckout TIMESPEC

Specifies the long checkout trigger threshold for the specified feature. Set to 0 to disable.

ConfigureFeature FEATURE -userlongcheckout USER TIMESPEC

Specifies a feature and user-specific long checkout trigger threshold for the specified feature. Set to 0 to disable.

ConfigureFeature FEATURE - mincap INTEGER	Specifies a minimum capacity for the feature to be considered for the saturation notification.
ConfigureFeature FEATURE - triggerperc INTEGER	Specifies the saturation trigger percentage threshold for the specified feature.
ConfigureFeature FEATURE - triggeruse INTEGER	Specifies the saturation trigger concurrent usage number threshold for the specified feature. This setting takes precedence over the triggerperc setting.
ConfigureFeature FEATURE - spanalltags	Consider all tags when calculating percentage and concurrent use for saturation check. This setting is not compatible with features specified in TAG/FEATURE form.

Example TIMESPEC values are: 10s, 5m, 3h, 20d, 4w, and 1y. If specifying multiple configuration items for a single object, the items can be combined in the same line.

Security

User Management

Monitor manages users, but not passwords. Instead, the server can authenticate users against an LDAP system, or it can use the operating system as an authentication proxy.

The latter means that any user who can authenticate with the operating system can also log in to the Monitor interface with the same user name. The details of this process differ per platform, as described below.

Additionally, VovUserGroups can be created and used to assign roles. For example the `vovusergroup` utility can be utilized to create a VovUserGroup called "mygroup" that is based off a UNIX group, or LDAP group of the same name.

UNIX

```
% vovusergroup populate mygroup -unix mygroup
```

LDAP

```
% vovusergroup populate mygroup -ldap mygroup
```

This can be run manually, or more practically as a scheduled task. After the "mygroup" VovUserGroup exists, it can be assigned a security role as shown for the "queuemgrs" group in the Security Principles section.

LDAP Authentication

To enable LDAP authentication, first configure the LDAP interface as described in the [LDAP Integration](#) section of this manual. After the configuration has been performed and tested, configure the product server to attempt authentication via LDAP by adding this line to the `licmon.swd/policy.tcl` file:

```
set config(enableLdap) 1
```

Once the line has been added, reread the change by resetting Monitor via the **System** page under the **Admin** tab, or by issuing the following commands at the CLI:

```
% vovproject enable licmon  
% vovproject reread
```

OS-based Authentication on UNIX

When Monitor is running on UNIX, the pluggable authentication module system (PAM) is used. PAM is used in most situations and can be configured to authenticate against NIS or LDAP. If PAM is not configured on the server machine or the user is not successfully authenticated by PAM, the server will attempt to retrieve the password hash from the local password database and compare that against what is entered as the password in the web browser. This requires a utility to be present that can access the password database. The utility is created by running the following commands as root:

```
% cd $VOVDIR/..  
% ./scripts/SETTASKERUID.csh
```

Again, this authentication mode is only used when PAM cannot service the authentication request and at no time is the password decrypted from its hashed form that is found in the local password database.

OS-based Authentication on Windows

When Monitor is running on Windows, the Windows API is used. The API supports both local and domain users. To log in as a local user, simply enter the user name. To log in as a domain user, use the format of domain/user. Note that Windows allows for mixed-case authentication (ie joe = Joe = JOE). This requires that the case for the security principle and/or tag data access configurations match the user name that is being used to log in. See below for more details on configuring security principles and tag data access.

Security Principles

Every user possesses a security principle.

By default, the user that starts the Monitor instance is granted the ADMIN security principle and all other authenticated users possess the USER principle. The ADMIN security principle is required for a user to access the Admin tab of the web interface. To change the security principle for a user, modify the `licmon.swd/security.tcl` file. The syntax for the security configuration line is:

```
# This is licmon.swd/security.tcl  
vtk_security <username>|-group <vovusergroup> <principle> <hostlist>
```

Available principles are USER and ADMIN. The hostlist can be a single host name, a list of hosts, or a + wildcard character to represent all hosts. The host specification controls which hosts the user is allowed to have the associated security principle from. With regards to the web interface, this will always be the Monitor server machine. The CLI, however, allows for remote connectivity that can be used to perform Monitor system administration. The host specification provides granular control over the user/host combinations that are allowed to perform CLI administration commands.

In addition to users, VovUserGroups that exist in the system may be specified in the security.tcl file. These groups are managed with the `vovusergroup` utility, and can be derived from unix groups, LDAP, or user lists. More information about them can be found on the VovUserGroups page. In the below example, the VovUserGroup called "queuemgrs" is given admin rights when logging in from the IP address range shown.

```
# This is licmon.swd/security.tcl
vtk_security cadmgr      ADMIN +
vtk_security joe         ADMIN +
vtk_security -group queuemgrs ADMIN 192.168.10.1-192.168.10.55
vtk_security +           USER +
```

Windows Domain Accounts

If the user who starts the Monitor instance is a domain account, the security configuration may need to be adjusted to specify the domain account as part of the user name. For example, if the authentication domain is "MYDOMAIN" for user "joe", the security configuration for ADMIN would need to resemble:

```
# This is the security.tcl file.
vtk_security "MYDOMAIN/joe" ADMIN +
```

The same should be applied for all other security principle assignments for Windows domain accounts.

Register Security Changes

When changes are made to the security configuration, the server must be instructed to read in the changes. This is accomplished by resetting Monitor via the **System** page under the Admin tab or with the following CLI commands:

```
% vovproject enable licmon
% vovproject reread
% vovproject sanity
```

Control Access to Tag Data

There may be situations where company policy may not allow all data to be visible to all users that have access Monitor. This section describes how Monitor can be locked down so that users only have visibility to the data for which he or she is authorized to view. This is made possible by the subsystem access control list (ACL) functionality.

This functionality is also used to ensure visibility to tags that are not actively being monitored via sampling, as described in [General Monitoring Configuration](#). This would typically be achieved using the EVERYBODY keyword described below.

Configuring

Tag access is defined in the `licmon.swd/vovlmd/config.tcl` file using the `setTagAccess` procedure. This procedure takes two arguments:

- A single tag or a list of tags.
- A single user name or a list of user names that are to be granted access to the tag being configured. The ADMIN keyword can be used to grant access to all configured administrators. The EVERYBODY keyword can be used to grant access to all users. This is the default behavior when the tag is being actively monitored via sampling.

Example:

```
#  
# This is a fragment of vovlmd/config.tcl  
#  
setTagAccess IN_MGC "gupta sandeep mario"  
setTagAccess IN_SNPS "gupta sandeep mario"  
setTagAccess EU_MGC "franz javier"  
setTagAccess EU_CDN "javier oliver"  
setTagAccess IN_CDN EVERYBODY  
setTagAccess "CN_MGC CN_SNPS" "wchen"
```

If visibility of a tag needs to be restored to all users, the procedure must be called with the EVERYBODY keyword.

Security Analysis

Monitor also provides a security analysis report that aids the administrator in further locking down the various pieces of data that may contain sensitive information.

This report is available via **Admin > System > Security Information**. The report highlights items that are not secure and provides instructions for securing them. This functionality is only available if the Monitor server is running on UNIX.

System Information

Details about the Altair Monitor system can be found below.

Tag Access

To control access to tags on a per-user basis, use the `setTagAccess` procedure in the `vovlmd/config.tcl` file (see security documentation in the administrator's guide for more information on this). This feature is optional, used only when company policy prevents usage data from being generally available. Below is a list of the daemons that are currently in server memory, along with their security status. The daemon ID can be clicked to display the detailed access control policy for that daemon. This is normally used for debugging purposes only.

TAG ID	Security Status
--------	-----------------

ALM 000041622	open to all users
---------------	-----------------------------------

Security Analysis

Controlling tag access is one part of securing the data that is present in the Altair Monitor system. The below shows a list of potential areas that may contain sensitive data, along with their security status. It is recommended to learn about these areas to determine if they should be secured before taking any action.

- ✗ **This platform, i.e. win64, does not fully support security.**
Switch to a Unix/Linux platform for full support.
- ✗ **Guest access is enabled on port 5556.**
Set `config(readonlyPort)` to 0 in the `policy.tcl` file to disable.
- ✗ **HTTP file access is enabled.**
Set `config(disablefileaccess)` to 2 in `policy.tcl` file.
- ✗ **Alerts are Open**
Run:
% `vovproject enable licmon`
% `vovfsgroup create /system/aclddef/alerts`
% `vovfsgroup acl /system/aclddef/alerts APPEND USER denyby VIEW`
[Or click here to close Alerts.](#)
- ✗ **Event Stream is Open.**
Run:
% `vovproject enable licmon`
% `vovfsgroup create /system/aclddef/notify/start`
% `vovfsgroup acl /system/aclddef/notify/start APPEND USER denyby VIEW`
[Or click here to close Event Stream.](#)

Figure 37: Tag Information

View and Edit ACLs

As previously mentioned, the underlying mechanism that provides these security measures is provided by the subsystem ACL functionality. To view and/or manually edit ACLs for a license daemon object, visit the **Security Information** page and click on one of the license daemon object IDs at the bottom of the page. The resulting ACL control page is meant for advanced users who are familiar with the Access Control List system.

Access Control List (ACL) Management

This page manages the access control list for a system object. System objects include jobs, files, license daemons, and FairShare groups.

Managing ACL for license daemon: 000041622

Current ACL

Agents and actions for an ACL cannot be changed. Instead, create a new ACL by adding a new allowed action to the ACL for the OWNER agent using the form below.

Agent	Name (optional)	Actions	Operations
OWNER		EDIT VIEW FORGET EXISTS	Undeletable
ADMIN		EDIT VIEW FORGET EXISTS	Delete
EVERYBODY		VIEW EXISTS	Delete

Showing 3 out of 3 rows | Limit rows to display: 10

Reset Acl

Add Allowed Action to ACL

Agent

Name (optional)

Actions

EVERYBODY

☒ EXISTS ☒ EDIT ☒ VIEW ☒ FORGET ☒ DELEGATE

Add

Current Security

You are denyby from uslt70 with security 5
ACL security = ADMIN
ACL key = aclid
ACL host = USLT70
ACL result = ok
ACL agent = USER
ACL user = denyby

Figure 38: Access Control Policy

Web Server Configuration

HTTP Access Models

There are 3 HTTP access models:

- Legacy
- Internal/External
- Nginx

Legacy Webserver

The Legacy webserver is the basic web server that is internal to vovserver and serves content directly to web browser clients.

All traffic is transmitted using HTTP protocol and is unsecured. This method is appropriate for REST versions up to version 2.0.

This is the case when

- `webport=0`, or
- `webport != 0` and `webprovider=nginx`

Internal Webserver

The Internal webserver is an enhanced web server that is internal to vovserver, for secure pages and all REST versions.

The Internal webserver is established when

- `webport != 0` and `webprovider=internal`

To specify the web port at product start, refer to the product-specific documentation for startup. To change the port in an already-running product instance, see *Advanced Control of the Product Ports*. To enable SSL support (HTTPS), follow the steps in [Configure the TLS/SSL Protocol](#).

You get REST v3 API support from this webserver, and we still transparently delegate some HTTP requests to the old web server on the VOV port.

The Internal server securely handles all incoming traffic, decrypting it before handing it off to the locally running vovserver. Likewise, any response that is sent back to the browser is routed through the Internal webserver, which encrypts the response and sends it to the browser. This implementation is known as an SSL termination proxy.

nginx Webserver

The vovserver serves content to a proxy webserver (nginx), which communicates to web browser clients. Under this model, SSL can be enabled, securing all traffic using the HTTPS protocol.

The nginx web server is enabled when the web port is configured with a non-zero value. To specify the web port at product start, refer to the product-specific documentation for startup. To change the port in an already-running product instance, see *Advanced Control of the Product Ports*. To enable SSL support (HTTPS), follow the steps in [Configure the TLS/SSL Protocol](#).

For experts only, advanced customizations to the nginx configuration can be made by modifying its configuration template. Configuration templates are searched for in the following locations:

Order	Type	Path
1.	Instance-specific	<code>\$SWD/vovnginxd/conf/nginx.conf.template</code>
2.	Site-wide	<code>\$VOVDIR/local/config/vovnginxd/nginx.conf.template</code>
3.	Installation-specific(edits not recommended)	<code>\$VOVDIR/etc/config/vovnginxd/nginx.conf.template</code>

If customizations are intended, it is recommended to start with a copy of the default configuration template shown at location 3 above and place into either location 1 or 2.



Note:

- The configuration template is copied into the nginx configuration directory located at `$SWD/vovnginxd/conf`, named as `nginx.conf`. The copy is made upon product start, as well as any time the web port or SSL configuration is changed.
- Changes to the actual configuration file can be read into nginx via the `vovdaemonmgr reread vovnginxd` command, but such changes will be overwritten the next time the configuration template is copied.
- The configuration template contains keywords surrounded by @ signs, such as `@WEBPORT@`, that are dynamically substituted with values during the copy process. Removal of these keywords is not recommended, as it may effect the ability for nginx to be reconfigured in the event of a vovserver failover.

Configure the TLS/SSL Protocol

The internal and nginx webserver support TLS/SSL Protocol communication via "https" - prefixed URLs when configured correctly.



Note: TLS encryption is enabled for client communication to the VOV port by default starting in 2025.1.0. TLS can be disabled by setting the new `comm.tls.enable` server configuration parameter to 0. Older version clients that do not use TLS will be allowed to connect without TLS encryption.

The vovserver serves content to a proxy webserver (nginx), which communicates to web browser clients. Under this model, SSL can be enabled, securing all traffic using the HTTP protocol.

When SSL is enabled, nginx will look for an SSL certificate/key pair in the following locations:

Order	Type	Path	Files
1.	Site-wide wildcard	\$VOVDIR/local/ssl	wildcard-crt.pem wildcard-key.pem
2.	Host-specific	\$SWD/config/ssl	hostname-crt.pem hostname-key.pem
3.	Host-specific (auto-generated and self-signed)	\$SWD/config/ssl	hostname-self-crt.pem hostname-self-key.pem



Note:

- For hostname, use the actual host name that will be used to access the web UI. This will be the value of VOV_HOST_HTTP_NAME that was set in the configuration. If not defined, the value of VOV_HOST_NAME is used instead.
To use the fully qualified domain name, the value of VOV_HOST_HTTP_NAME must be set.
- Self-signed certificates will present security warnings in most browsers.

Updating the TLS/SSL cert requires restarting the webserver so that the cert files can be re-read. For the internal webserver, see, "Restarting the Webserver" below.

Guest Access Port

The vovserver can be configured to enable a guest-access port, also called the read-only port due to the limited privileges allowed by the port. This port bypasses the login prompt and provides the user with a READONLY security principle, which disallows access to writable actions as well as certain pages in the UI.

To specify the guest access port at product start, refer to the product-specific documentation for startup. To change the port in an already-running product instance, follow the steps in *Advanced Control of the Product Ports*.

Transition from nginx Webserver to Internal

To transition from external (nginx) to the internal web server, follow these steps:

1. Shut down nginx with the command `vovdaemonmgr stop vovnginxd`.
2. Delay for 5 seconds with the command `sleep 5`.

3. Start the internal web server with `vovservermgr config webprovider internal`.

Restarting the Webserver

Complete the following steps to restart the webserver without bringing down vovserver.

1. Enter the following:

```
vovservermgr config webport 0
```

2. Wait five seconds, then enter:

```
vovservermgr config webport $VOV_WEB_PORT_NUMBER
```

Web Interface

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

```
### Global checkout data origin to use for historical reports
# Default is "samples"
# Legal values are:
#   "samples"  lmstat
#   "logs"     debug logs
#   "merged"   merger of samples and debug logs
# set VOVLM(origin) "samples"

### CSV export
# Set delimiter character for CSV exports (for tab use "\t")
# set VOVLM(delimiter, csv) ",", "

### Custom footer message, printed above Altair Engineering copyright statement
# VovCGI::setFooter "Company Confidential"

### Adding a custom tab/menu/page
# VovCGI::addTab <tab> <url> [options]
# VovCGI::addMenu <menu> <url> [options]
# VovCGI::addPage <page> <url> [options]
#
# Tabs/menus/pages that contain a space must be enclosed within double quotes.
#
# Options:
# -access <rule>      Set tab/menu/page visibility and access, see access rule
#                      section below for more information.
# -autorefresh <0|1>  Enables autorefresh and control, for menus only.
# -class <class>      Show a page type icon next to the page link, for
#                      pages only. Classes are: heatmap, histogram, plot,
#                      statplot, and table.
# -disabled <0|1>     Disables the tab/menu/page link and displays it with
#                      strikethrough.
# -msg <txt>          Message to display for disabled items.
#
```

```
# Example of adding a custom tab with menu and pages:
# VovCGI::addTab "My Tab" "/cgi/mycgi.cgi" -access "user=joe"
# VovCGI::addMenu "My Menu 1" "/cgi/mycgi.cgi?option=1" -autorefresh 1
# VovCGI::addPage "My Page 1" "/cgi/mycgi.cgi?option=2" -class table
# VovCGI::addPage "Google" "http://www.google.com"
#
# Custom CGI scripts can be placed into either the $VOVDIR/local/cgi directory
# (for all products) or the $SWD/cgi directory (for LM specifically).
#
# Example of adding a custom menu to an existing tab:
# VovCGI::setTab "Current"
# VovCGI::addMenu "My Menu 2" "/cgi/mycgi1.cgi?option=1" -autorefresh 1
#
# Example of adding a custom page to an existing menu:
# VovCGI::setMenu "Current" "Checkouts"
# VovCGI::addPage "My Page 2" "/cgi/mycgi2.cgi?option=1" -class table

### Changing an existing tab/menu/page
# VovCGI::setTabOption <tab> <option> <value>
# VovCGI::setMenuOption <tab> <menu> <option> <value>
# VovCGI::setPageOption <tab> <menu> <page> <option> <value>
#
# Options and possible values are listed in the previous section.

### Web UI tab/menu/page/component visibility and access
# VovCGI::setTabOption <tab> access <access-rule>
# VovCGI::setMenuOption <tab> <menu> access <access-rule>
# VovCGI::setPageOption <tab> <menu> <page> access <access-rule>
# VovCGI::setComponentOption <component> access <access-rule>
#
# Tabs/menus/pages/components that contain a space must be enclosed within double
# quotes.
#
# Access rules:
# 1. role=<role>
#    Roles are: admin | leader | user | readonly
# 2. group=<group1>,<group2>,<groupN>...
#    Groups are defined via the vovusergroup utility (see docs)
# 3. user=<user1>,<user2>,<userN>...
#    Users who are logged into LM
# 4. disabled
#    Disabled altogether, even admin role cannot access.
#
# Enclose multiple access rules withing double quotes, separated by spaces.
# The most restrictive access rule wins.
#
# Components:
#   homeutil - the current utilization section on the home page
#   homewait - the current wait section on the home page
#   homeview - the my view section on the home page
#   homeservers - the current servers section on the home page
#   lmremove - the ability to use lmremove on the current checkouts page
#               default: user
#               note: if individual users are added, these users may also
#                     remove checkouts from other users
#   usernames - any display or control of usernames
#   emails - and display of email addresses
#
# Examples:
# VovCGI::setTabOption Current access "role=admin group=itmgrs"
# VovCGI::setMenuOption History Denials user=joe,mary
# VovCGI::setPageOption History Usage "Checkout Details" access group=grp1,grp2
# VovCGI::setComponentOption usernames access role=admin
```



```
#
# Note:
# Access applied to the Home > Alerts page affects the alerts status area in
# the page banner at the top of the screen.

##### Home page view settings
# Establish a default view for all users who do not have one
# set VOVLM(view,default,owner)  lmadmin
# set VOVLM(view,default,name)  globalview
#
### Set the number of rows shown in the view on the home page
# set VOVLM(view,default,limit) 200
#
# Set the default sorting for the view on the home page
# Default is "inuse"
# Available sort values are:
#   "name"      - the filter name
#   "tags"      - matched tag(s)
#   "features"  - matched feature(s)
#   "inuse"     - current number in-use
#   "total"     - total capacity available
#   "percent"   - current percentage of in-use/capacity
# set VOVLM(view,default,sortby) total
#
### Set default sorting direction for the view on the home page
# Leave commented out or set to 0 for descending, set to 1 for ascending
# set VOVLM(view,default,ascend) 1

##### Mailto links
### Set delimiter character for e-mail addresses in mailto links
# set VOVLM(delimiter,emails) ", "
#
### Set domain suffix for mailto links
# set VOVLM(domain) "mycompany.com"
#
#
### Use LDAP for e-mail addresses instead of user name
# set VOVLM(ldapEmail) 1
#
# Set default subjects and bodies
# set VOVLM(subject,current,checkouts,individual) ""
# set VOVLM(body,current,checkouts,individual) ""
# set VOVLM(subject,current,checkouts,summary) ""
# set VOVLM(body,current,checkouts,summary) ""
# set VOVLM(subject,current,features,individual) ""
# set VOVLM(body,current,features,individual) ""
# set VOVLM(subject,current,features,summary) ""
# set VOVLM(body,current,features,summary) ""
# set VOVLM(subject,history,checkoutstats,individual) ""
# set VOVLM(body,history,checkoutstats,individual) ""
# set VOVLM(subject,history,checkoutstats,summary) ""
# set VOVLM(body,history,checkoutstats,summary) ""
# set VOVLM(subject,history,checkoutdetails,individual) ""
# set VOVLM(body,history,checkoutdetails,individual) ""
# set VOVLM(subject,history,denialstats,individual) ""
# set VOVLM(body,history,denialstats,individual) ""
# set VOVLM(subject,history,denialstats,summary) ""
# set VOVLM(body,history,denialstats,summary) ""
# set VOVLM(subject,history,denialdetails,individual) ""
# set VOVLM(body,history,denialdetails,individual) ""
# set VOVLM(subject,history,usagetrends,individual) ""
# set VOVLM(body,history,usagetrends,individual) ""
# set VOVLM(subject,history,usagetrends,summary) ""
```

```
# set VOVLm(body,history,usagetrends,summary)      ""
```

LDAP Integration

Monitor includes an interface to LDAP that can provide authentication services, extract user and display user attributes throughout the various pages in the web interface, and drive custom groups of users for reporting purposes.

Configure the LDAP Interface

To configure the interface to LDAP, first copy the configuration template, \$VOVDIR/etc/config/ldap/ldap.cfg, to the licmon.swd/config directory. This configuration template is self-documented, with every possible configuration item explained:

Example Configuration: \$VOVDIR/etc/config/ldap/ldap.cfg

```
# LDAP interface configuration file.
# Should be placed in the SWD/config directory.
# All settings are required unless specified otherwise.
# Unused optional settings should be commented out.

#####
###SERVER###
#####
#Server host that is running LDAP.
# Required.
set LDAP(host)      ldapsrv

# Port through which LDAP communication occurs.
# Optional, defaults to 389.
set LDAP(port)      389

# SSL.# Set to 1 if LDAP server requires SSL to connect.
# Set to 0 if SSL is not required.
set LDAP(ssl)              0

# Search base. Required. The highest level of the LDAP data information tree
# (DIT) that contains the information sought by the RTDA application.
set LDAP(base)      dc=my,dc=company,dc=com

# LDAP Binding Distinguished Name
# If the LDAP server allows anonymous connections (binding), then the bindDN
# line should be commented out. Otherwise, binding to the LDAP server
# requires this correctly configured bindDn string to use in concert with the
# bind # password. The bind password is specified elsewhere -- via
# "vovldap_setup setpassword".
# An LDAP "distinguished name" contains a list of key-value pairs similar to
# the template shown here.
set LDAP(bindDn)      cn=ldapmgr,ou=People,dc=my,dc=company,dc=com

#####
#### PEOPLE ###
#####
#People search base. Optional, speeds up searches by narrowing them to LDAP
# entries that are for people based on their tree location. This is prepended to
# the LDAP base specified above. If all users are not located in the tree
# location, comment-out this setting.
set LDAP(peopleBase)  ou=People
```

```
# People object class. Optional, speeds up searches by narrowing them to LDAP
# entries that are for people based on their object class. This is added to the
# filter specification when searches are requested.
set LDAP(peopleSearchObjectClass) account

# Relative distinguished name (RDN) used to search for users. Required.
# This should be set to the name of the LDAP attribute that contains the unique
# login name that is used by the operating system.
set LDAP(peopleSearchRdn) uid

# Attributes that are returned from search and displayed on the user LDAP
# information page (reachable by clicking the LDAP folder icon in the web UI).
# Required.
set LDAP(peopleReturnAttrib) {cn uid}

# Attribute ordering and mapping to human-readable headings for display in the
# user LDAP information page. Optional. If commented-out, attributes and values
# will be returned as they are named and ordered in LDAP.
set LDAP(map) { {cn AS Name} {uid AS "User ID"} }

# Attribute that contains the user e-mail address. Optional. Switches mailto
# links in the web UI to use the LDAP email address as opposed to using the user
# name. This setting is only used if this functionality is configured in the
# SWD/config/web.cfg file.
set LDAP(emailAttribute) mail

#####
#### GROUPS ####
#####

# Group search base. Optional, speeds up searches by narrowing them to LDAP
# entries that are for groups based on their tree location. This is prepended to
# the LDAP base specified above. If all groups are not located in the tree
# location, comment-out this setting.
set LDAP(groupBase) ou=Groups

# Group object class. Optional, speeds up searches by narrowing them to LDAP
# entries that are for groups based on their object class. This is added to the
# filter specification when searches are requested.
set LDAP(groupSearchObjectClass) groupOfNames

# Relative distinguished name (RDN) used to search for groups. Required. This
# should be set to the group attribute that contains the name of the group.
set LDAP(groupSearchRdn) cn

# Attribute used to denote a user as a group member. Required. This should be
# set to the repeated group attribute that contains a member name.
set LDAP(groupMemberAttrib) member

#####
#### CUSTOM GROUPS ####
#####

# Define LDAP attributes and/or groups to use for populating group memberships
# in the LM DB. These definitions are read by "vovldap_update_memberships",
# which is called once every week by the "live_lm_update_memberships_from_ldap"
# liveness script. These definitions establish a group membership starting at
# the time LDAP is is queried and ending 10 years later. The formats are:
# set LDAP(attribute,<customGroupType>) <ldapAttributeName>
# set LDAP(groups,<customGroupType>) {<ldapGroupName1>, <ldapGroupName2>,...}
# set LDAP(attribute,location) l
# set LDAP(attribute,department) ou
# set LDAP(groups,division) {Engineering, Maintenance}
```

```
#####  
#### FAILOVER ####  
#####  
# Number of additional servers that should be contacted if the primary server  
# is either down or does not contain the information being sought. Failover  
# is used solely for authentication services.  
# set LDAP(failoverServers) 1  
# Copy and modify any of the primary settings above to match that of each  
# failover server. Any primary setting that is not redefined below will be used  
# in the failover configuration as well. For each primary setting that is  
# overridden, use LDAP_FO_N(X) instead of LDAP(X), where N is the failover  
# server number and X is the setting. Example, where the primary server is  
# ldapsrv1, the failover servers would be:# set LDAP_FO_1(host) ldapsrv2  
# set LDAP_FO_2(host) ldapsrv3  
# EOF
```

Set the Bind Password

If anonymous binding is not allowed by the LDAP system, a bind account must be defined in the configuration and a password must be set. A utility is provided that is used to store the LDAP bind account password inside the Monitor server. This password will be used any time LDAP is interacted with. The utility is `vovldap_setup` and is called in this manner:

```
% vovproject enable licmon  
% vovldap_setup setpassword secret
```

The utility can also be used to show the existing password as well as the overall configuration that has been established in the configuration file referred to above. The complete usage syntax for this utility is:

```
vovldap_setup: Usage Message  
  
A utility to manage certain LDAP integration configuration items.  
  
USAGE:  
    % vovldap_setup [OPTIONS]  
  
OPTIONS:  
    -h                                -- This help.  
    -v                                -- Increase verbosity. Repeatable.  
    setpassword <passwd>              -- Password for user to bind with  
                                        primary server.  
    setfopassword <#> <passwd>        -- Password for user to bind with  
                                        failover server(s).  
    showconfig                        -- Show the current configuration.  
    showpasswords                     -- Show the current passwords.  
  
EXAMPLES:  
    % vovldap_setup showconfig  
    % vovldap_setup setpassword secret  
    % vovldap_setup setfopassword 1 secret
```

Once the configuration file is in-place and the bind password is set (if applicable), the LDAP interface will be activated, enabling the LDAP folder icon links in the web interface. However, for the LDAP connection to work, the settings in the configuration file must be modified to match that of the LDAP system that is being connected to.

Important: If a bind password is required, once it is configured with `vovldap_setup`, the `LDAP_BIND_PW` property on object 1 is visible to everyone.

Authentication

Refer to [Security](#) for details on configuring LDAP authentication.

Display User Information

The presence of the LDAP configuration file mentioned above enables the LDAP icon to appear anywhere a user name is printed in the various report pages in the web UI. This icon, when clicked, will extract the attributes specified in the LDAP configuration file and display them in a pop-up window. The `peopleReturnAttrib` setting shown in the configuration example above specifies which attributes to return. The `map` setting allows for renaming the attributes into user-friendly names, as well as specifying the display order of the attributes.

Obtain E-mail Addresses

Some reports in Monitor contain a mail icon that allows for quickly e-mailing a user or group of users. By default, this icon represents a `mailto` link that is composed of the user name as reported to Monitor by the license server manager being monitored. This link can also be populated with the e-mail address populated in LDAP for the specified user by defining the e-mail attribute that is used by LDAP. Additionally, the following line must be set in the [web interface configuration](#) to enable the lookup to be performed in the web interface:

```
set VOVLm(ldapEmail) 1
```

Obtain Custom Group Definitions From LDAP

In addition to displaying user information found in LDAP, Monitor can also utilize LDAP attributes for defining custom group definitions. In the example above, the last configuration lines specify that:

- The "l" LDAP attribute is to be used to map users into the custom group entitled "location".
- The "ou" LDAP attribute is to be used to map users into the custom group entitled "department".
- Members from the "Engineering" and "Maintenance" LDAP groups are populated into the custom group entitled "division".

Refer to the [Project and Group Management](#) section of this manual for details on how to extract and populate custom group definitions using LDAP.

Test LDAP

A utility, `vovldap_query`, has been provided that can be used to test the connection to LDAP, as well as lookup attributes at the command line. The utility's usage syntax is:

```
vovldap_query: Usage Message

SYNOPSIS:
    Utility to perform LDAP lookups.

USAGE:
```

```
% vovldap_query <ARGUMENTS> [OPTIONS]

ARGUMENTS:
  da <dn>                -- Get attributes for specified DN.
  dn <user>              -- Get DN for specified user.
  gm <group>             -- Get members for specified group.
  ua <user>              -- Get attributes for specified user.
  ug <user>              -- Get groups for specified user.
  um <attribute=value>   -- Get users matching specified attribute
                        and value.

OPTIONS:
  -v                    -- Increase verbosity
  -q                    -- Quiet
  -attributes "names"  -- List of attributes to constrain da/ua
                        search result.
  -orderandmap          -- Order and map result per configuration file.

EXAMPLES:
% vovldap_query ua jchen
% vovldap_query ua jchen -orderandmap
% vovldap_query ua jchen -attributes "cn mail"
% vovldap_query um "l=New York"
```

Workweek Definition

Monitor includes support for workweek filtering for certain reports. The workweek defaults to 08:00-18:00 (8am to 6pm) Monday - Friday.

Using the browser interface, visit the **Workweek** administration page. On this page one will find the current workweek definition. Users logged in with 'admin' privileges will be able to redefine the workweek definition by clicking **Edit Workweek Definition**.

System Information

Details about the Altair Monitor system can be found below.

Current Workweek: 08:00 - 18:00 (weekend = Sat, Sun)

Workday range:

08:00

18:00

Weekend: ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☒ Sat ☒ Sun

Save

Cancel

Figure 39: Workweek Information

Command Line Operations

Sample a License Server

The Monitor CLI provides the ability to manually sample a license server to gather availability and utilization data. This can be helpful in debugging problems with the connectivity to a license server, or the data that is returned from the license manager status command.

Operation

To manually sample a license server, regardless of the license manager type, first setup the CLI:

```
% vovproject enable licmon
```

After setting up the CLI, the license manager-specific sampling command must be executed. The available sampling commands are:

- [ftlm_parse_beta](#)
- [ftlm_parse_clearcase](#)
- [ftlm_parse_dsls](#)
- [ftlm_parse_elas](#)
- [ftlm_parse_flexlm](#)
- [ftlm_parse_hasp](#)
- [ftlm_parse_icad](#)
- [ftlm_parse_ghs](#)
- [ftlm_parse_gns](#)
- [ftlm_parse_lmx](#)
- [ftlm_parse_lum](#)
- [ftlm_parse_lum_denials](#)
- [ftlm_parse_lstc](#)
- [ftlm_parse_mathlm](#)
- [ftlm_parse_rlmstat](#)
- [ftlm_parse_rtdakey](#)
- [ftlm_parse_sentinel](#)
- [ftlm_parse_silvaco](#)
- [ftlm_parse_olicense](#)
- [ftlm_parse_qftest](#)
- [ftlm_parse_tsystems](#)
- [lm_parse_remotelm](#)

Click the links above to learn the arguments required and options supported for each command.

Parse a Debug Log

A need may arise to manually parse a debug log to gather denial and/or utilization data. This can be helpful in debugging problems with the data obtained from the file, or with parsing the file itself.

For parsing FlexNet Publisher debug logs in particular, be sure to first review [Manage Data from Different Origins](#). A number of caveats exist when importing debug logs. It is important to be aware of these ahead of time to set expectations, and better plan for how data will be utilized.

With FlexNet Publisher debug logs in particular, some major questions to be considered:

- Are the debug logs static or regularly rotated?
- Are you attempting to parse the log for checkouts as well as denials?
- Do you need to merge checkouts from multiple origins within the same time frame?
- Do you wish to access imported log data on the same graph/report as sampled data?
- Do you need capacity information for checkouts loaded from debug logs?
- Do you need to access imported data live, or just for archive purposes?

Operation

To manually parse a debug log, first setup the CLI by enabling the Monitor project:

```
% vovproject enable licmon
```

After setting up the CLI, the debug log parsing command must be executed. As of this version, the following debug log parsers are provided:

- Altium: [ftlm_parse_altiumlog](#)
- CodeMeter: [ftlm_parse_codemeterlog](#)
- Dassault: [ftlm_parse_dsIslog](#)
- FlexNet Publisher: [ftlm_parse_debuglog](#)

Click each link above to learn the arguments required and options supported for the command.

Load the Database

A need may arise to manually load checkout or denial data into the database. This will be required if importing historical checkout data that is obtained from parsing a debug log, or if reloading data for checkouts and/or denials. It can also be helpful in debugging database loading problems.

1. To manually load the database, first setup the CLI:

```
% vovproject enable licmon
```

2. Change to the directory that contains the data that is to be loaded. This data is stored in the checkouts/denials data files, located in `licmon.swd/data` under a respective subdirectory for each data type. The data is organized into files that are named for each day that contains data.
3. After setting up the CLI, the database loader command must be executed. The available commands are:

- [vovsql_load_checkouts](#)
- [vovsql_load_denials](#)

4. Click the links above to learn the arguments required and options supported for each command.

Set Feature Capacity

Monitor tracks the capacity of licenses in addition to utilization data. This is done so automatically with a periodic job, via the `SNAPSHOT_CAPACITY` task by loading a snapshot of the capacity of all features every hour by default, comparing the values against what is in the database and if they differ, inserting a new capacity value for each feature. In parallel, a data file also maintains a history of the capacity changes. This data file is useful for multi-site situations to aggregate capacity data to a parent site.

The `ftlm_capacity` utility is used to show and manage the capacity values in the Monitor database. The utility provides the following functions:

- Show existing capacity values
- Reload capacity values
- Set capacity value for a specific feature for a specific time

Operation

To execute `ftlm_capacity`, first setup the CLI:

```
% vovproject enable licmon
```

After setting up the CLI, the command can be executed:

```
ftlm_capacity: Usage Message

    Manages feature capacity information in the Monitor database.

SYNOPSIS:
% ftlm_capacity <ACTION> [OPTIONS]

ACTIONS:
show                -- Show latest capacity information.
dump                -- Same as show, only with UNIX timestamps.
clear               -- Clear capacity data from database.

checkmismatch       -- Check mismatch between latest DB capacity and live
                     data. Requires a tag to be specified with -tag.
fixmismatch         -- Fix mismatches that are found with checkmismatch.
                     Requires a tag to be specified with -tag.

load                -- Load information about capacity by parsing the
                     resources logs (in .swd/data/resources/)
snapshot            -- Snapshot capacity info using current information.
snapfromfile <FILE> -- Load snapshot capacity info from file. This action
                     does not work with the tag/feature/start/finish
                     options below.
                     Format of the file: setCapacity TAG FEATURE
                     TOKENS TIMESTAMP
set <TAG> <FEATURE> <TOKENS> <DATE_SPEC>
```

```

-- Set a new data point in the capacity table.
setforward <TAG> <FEATURE> <TOKENS> <DATE_SPEC>
-- Same as 'set' but delete all points where the
  timestamp is greater than the specified timestamp.

OPTIONS:
-v                -- Increase verbosity.
-all             -- For show and dump, display all capacities as
                  opposed to the latest only.
-n              -- Dry-run mode for write operations.
-tag            <TAG>    -- Constrain to a specific tag.
-feature        <FEATURE> -- Constrain to a specific feature.
-start          <DATESPEC> -- Specify start for load, show, and dump actions.
                  Default: equivalent to "1 year ago"
-finish         <DATESPEC> -- Specify finish for load, show, and dump actions.
-srcdir         <DIR>    -- Directory with the resources*log files.
                  Default: *.swd/data/resources
-wavedir        <DIR>    -- Directory where waves should be stored.
                  Default: *.swd/data/resources/waves
-force          -- Prevent redundant point detection for
                  snapfromfile action.

EXAMPLES:
% ftlm_capacity show
% ftlm_capacity dump
% ftlm_capacity show -feature hspice
% ftlm_capacity dump -tag EDA -feature hspice -start 20190215
% ftlm_capacity load
% ftlm_capacity load -tag EDA
% ftlm_capacity set EDA hspice 12 "1 year ago"
% ftlm_capacity setforward EDA hspice 12 "1 year ago"
% ftlm_capacity clear -tag EDA
% ftlm_capacity snapshot
% ftlm_capacity snapfromfile licmon.swd/data/capacity/2021.01.01.cap

```

Update Summary Table

Monitor utilizes summary tables for providing fast, high-level statistics for certain reports. The summary tables are automatically with a periodic job, via the UPDATE_SUMMARIES task by executing statistics gathering queries against the checkouts table and populating the results in the summary tables.

The ftlm_summary utility is used to update the summary tables at the CLI.

Operation

To execute ftlm_summary, first setup the CLI:

```
% vovproject enable licmon
```

After setting up the CLI, the command can be executed:

```
ftlm_summary: Usage Message

DESCRIPTION:
```

```
    Compute summary tables.

USAGE:
    % ftlm_summary [OPTIONS]
OPTIONS:
    -h                -- This message.
    -v                -- Increase verbosity.
    -start <TS>        -- Start time for checking duplicates
                        Default is one month ago.
    -end   <TS>        -- End time for checking duplicates
                        Default is 'now'.

    -full*            --
    -fullrange        -- Update all summaries in full start/finish range
                        in database

    -auto*            --
    -autorange        -- Look in metadata table to find out
                        range to update (the default)

    -tag      TagList    -- May be repeated
    -glitch   <timeSpec> -- Glitch removal control (default 20s)
    -cap      0|1        -- Cap usage at capacity (default 0)

EXAMPLES:
    % ftlm_summary -tag "EDA CDN"
    % ftlm_summary -start 20090101
    % ftlm_summary -tag SNSP -full
    % ftlm_summary -auto
```

VovUserGroups

VovUserGroups are an access control mechanism. They are internally-defined groups of users that are utilized to make the job of allocating access easier within Altair Accelerator tools.

VovUserGroups by themselves do not grant or restrict access. Once the group names are defined, however, they can be assigned security roles and thus make the job of administering security and access less time consuming.

The `vovusergroup` utility is used to manage VovUserGroups from the command line. This includes creating and deleting groups, appending them, and displaying group members.

VovUserGroups Utilization

VovUserGroups can be effectively utilized in a number of areas throughout the Altair Accelerator product suite. In all Altair Accelerator products, they can be used to easily set up access by mapping a VovUserGroup to a defined security role in the `security.tcl` file that resides in the Server Working Directory (SWD). VovUserGroups can also be used to assign access or restrictions to various components of the web interface.

Usage of vovusergroup utility:

vovusergroup: Usage Message

Utility to show, create and maintain "vovusergroup" structures based on user lists, Unix groups and LDAP groups.

USAGE:

```
vovusergroup <ACTION> <GROUP_NAME> <OPTIONS>
```

ACTIONS:

populate	Creates a new vovusergroup and populates it from a provided userList, an existing unix group or LDAP group. Any pre-existing vovusergroup with the same name is lost.
addusers	Creates a new vovusergroup (if necessary) and appends the provided userList.
removeusers	Removes the specified "userList" list of users from an existing vovusergroup.
delete	Removes a vovusergroup definition entirely, along with its list of users.
show	Displays the list of users in an existing vovusergroup.
list	Displays the defined vovusergroup names

SYNTAX:

```
vovusergroup populate <groupName> <userList>
vovusergroup populate <groupName> -unix <unixGroup>
vovusergroup populate <groupName> -ldap <ldapGroup>

vovusergroup addusers <groupName> <userList>
vovusergroup removeusers <groupName> <userList>

vovusergroup delete <groupName>
vovusergroup show <groupName>
```

ABOUT USER LISTS:

A "userList" above consists of usernames separated by spaces.
To generate a vovusergroup from LDAP, you must first have LDAP connectivity enabled (see Altair Monitor Administration Guide)

EXAMPLES:

```
vovusergroup populate chipA -ldap chipA_govusers
vovusergroup addusers myBlk jimc terryw suep ronaldb
vovusergroup show myBlk
```

VovUserGroups Support in Altair Accelerator Products

Currently, support for VovUserGroups is provided in the `security.tcl` file (in all products), and the `web.cfg` file (in Monitor only).

Query the vovserver

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

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

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

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

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

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

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

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

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

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

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

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

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

Examples

Queries you can run include the following:

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

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

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

For example, the following job submission

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

`vovselect` produces the following output:

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

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

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

For example, if the following job executes:

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

`vovselect` produces the following output:

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

vovselect

```
vovselect: Usage Message

Utility to query vovserver data.
```

USAGE:

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

OPTIONS:

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

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

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

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

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

SUPPORTED OBJECTS:

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

SUPPORTED FIELDS:

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

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

SUPPORTED FILTERS:

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

EXAMPLES:

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

Use vovselect for Querying

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

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

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

nc hosts	vovselect from TASKERS	vovselect from HOSTS
ARCH	ARCH	ARCH
CAPABILITIES	CAPABILITIES	NA
CAPACITY	CAPACITY	CPUS
CLASSRESOURCES	CLASSRESOURCES	NA
CLOCK	CLOCK	CPUCLOCK
COEFF	COEFF	NA
CONSUMABLES	CONSUMABLES	NA
CORES	CORESAVAIL	NA
CORESAVAIL	CORESAVAIL	NA
CORESTOTAL	CORESTOTAL	CPUS
CORESUSED	CORESUSED	NA
CPUS	CPUS	CPUS

nc hosts	vovselect from TASKERS	vovselect from HOSTS
CURLOAD	CURLOAD	NA
DOEXEC	DOEXEC	NA
DONETINFO	DONETINFO	NA
DOPROCINFO	DOPROCINFO	NA
DORTTRACING	DORTTRACING	NA
EFFLOAD	NA	NA
EXTRAS	EXTRAS	NA
FULLINFO	FULLINFO	NA
GROUP	GROUP	NA
HB	NA	NA
HBPP	NA	NA
HEARTBEAT	HEARTBEAT	NA
HOST	HOST	NAME
ID	ID	NA
IDINT	IDINT	NA
LASTJOBID	NA	NA
LASTUPDATE	LASTUPDATE	NA
LIFETIMEJOBS	LIFETIMEJOBS	NA
LOAD1	NA	NA
LOAD15	NA	NA
LOAD5	NA	NA
LOADEFF	NA	NA
MACHINE	MACHINE	MACHINE
MANUALPOWER	NA	NA
MAXLOAD	MAXLOAD	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
MESSAGE	MESSAGE	NA
MESSAGESYS	MESSAGESYS	NA
MESSAGEUSER	MESSAGEUSER	NA
MODEL	MODEL	NA
NAME	NAME	NAME
NUMJOBS	NA	NA
OSCLASS	OSCLASS	NA
PERCENT	PERCENT	NA
PERSISTENT	PERSISTENT	NA
PID	PID	NA
POWER	POWER	NA
RAM	RAM	NA
RAMFREE	RAMFREE	NA
RAMTOTAL	RAMTOTAL	RAMTOTAL
RAWPOWER	NA	NA
RELEASE	RELEASE	NA
RESERVEDBY	RESERVEDBY	NA
RESERVEEND	RESERVEEND	NA
RESERVEFORBUCKETID	RESERVEFORBUCKETID	NA
RESERVEFORID	RESERVEFORID	NA
RESERVEGROUP	RESERVEGROUP	NA
RESERVEJOBCLASS	RESERVEJOBCLASS	NA
RESERVEJOBPROJ	RESERVEJOBPROJ	NA
RESERVEOSGROUP	RESERVEOSGROUP	NA
RESERVESTART	RESERVESTART	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
RESERVEUSER	RESERVEUSER	NA
RESOURCECMD	RESOURCECMD	NA
RESOURCES	NA	NA
RESOURCESEXTRA	NA	NA
RESOURCESPEC	RESOURCESPEC	NA
RUNNINGJOBS	RUNNINGJOBS	NA
SLOTS	NA	NA
SLOTSTOTAL	SLOTSTOTAL	NA
STATSREJECTCORES	STATSREJECTCORES	NA
STATSREJECTOTHER	STATSREJECTOTHER	NA
STATSREJECTRAM	STATSREJECTRAM	NA
STATSREJECTRESERVED	STATSREJECTRESERVED	NA
STATSREJECTSLOTS	STATSREJECTSLOTS	NA
STATSVISITS	NA	NA
STATUS	NA	NA
SWAP	SWAP	NA
SWAPFREE	SWAPFREE	NA
SWAPTOTAL	SWAPTOTAL	NA
TASKERGROUP	TASKER	NA
TASKERNAME	TASKERNAME	NAME
TASKERSLOTSSUSPENDABLE	TASKERSLOTSSUSPENDABLE	NA
TASKERSLOTSSUSPENDED	TASKERSLOTSSUSPENDED	NA
TASKERSLOTSUSED	TASKERSLOTSUSED	NA
TASKERTYPE	TASKERTYPE	NA
TIMELEFT	TIMELEFT	NA

nc hosts	vovselect from TASKERS	vovselect from HOSTS
TMP	TMP	NA
TYPE	TYPE	NA
UPTIME	NA	NA
UPTIMEPP	UPTIMEPP	NA
USER	USER	NA
VERSION	VERSION	NA
VOVVERSION	VOVVERSION	NA

Reference

Troubleshooting

Before troubleshooting Monitor, it is recommended that you make yourself familiar with its theory of operation.

Components

Monitor is comprised of components that perform various functions. Refer to the table below.

Component	Description
vovserver	Controls all tasks. Receives and manages checkout data. Serves web pages.
vovtasker	The tasker (worker) that runs jobs to sample the license daemons and parse the resulting information.
vovlmd	Daemon responsible for setting up monitoring on configured license servers. vovlmd controls the processes that run to create sample data for each tag.
vovnotifyd	Daemon responsible for sending e-mails based on events that occur in the system or the licenses being monitored
vovtriggerd	Daemon responsible for archiving parsing job logs upon job failure. This daemon should remain off unless this feature is required to debug an intermittent monitoring failure. See the hints section below for more details.
vovnginxd	Daemon responsible for handling secure client connections and forwarding traffic to the vovserver. In Monitor, this daemon is on by default.

Daemon Control

The auxiliary daemons are normally launched at vovserver startup by the scripts in the `licmon.swd/autostart` directory. Check this if the auxiliary daemons are not starting properly. Example scripts are located in `$VOVDIR/etc/autostart`.

Daemon status can be checked and controlled, by visiting the **Daemons** page under the **Admin** tab, or by using the `vovdaemonmgr` command at the shell. The following table shows its usage information.

```
vovdaemonmgr: Usage Message

This is the command to show the status of daemons. You can also use this
command to start/stop the daemons.

NOTE: This command can only be used by the owner of the vovserver and on the
machine where the vovserver runs.

USAGE:
    vovdaemonmgr <SUBCOMMAND> [OPTIONS] [daemonsList]

SUBCOMMAND is one of:
    list      -- List the configured daemons. (list -all for all daemons)
    restart   -- Restart the specified daemons.
    show      -- Show the status of the specified daemons.
    status    -- Same as 'show'.
    start     -- Start the specified daemons.
    stop      -- Stop the specified daemons.

[daemonsList] is optional. When omitted, act on all daemons.

OPTIONS:
    -f          -- Force flag (for start only). If the start subcommand
                  has an explicit daemonsList, the specified daemons
                  will be started even if not configured.
    -h          -- Help usage message.
    -v          -- Increase verbosity.
    -retry <N>  -- For NIS, control how many retries to attempt, default 0
    -wait <N>   -- For NIS, control how long to wait between retries, default 0s

EXAMPLES:

% vovdaemonmgr list
% vovdaemonmgr list -all
% vovdaemonmgr status
% vovdaemonmgr status vovpreemptd
% vovdaemonmgr start vovnotifyd
% vovdaemonmgr start -force vovresourced
% vovdaemonmgr start -force -v -v vovresourced
% vovdaemonmgr stop vovlad
```

The main operations supported by the `vovdaemonmgr` command are:

- **status** - show daemon status
- **stop** - take down one or more daemons
- **start** - begin running one or more daemons

To determine the current situation, we recommend that you begin with `status` then use `stop/start` as needed.

Troubleshooting Hints

Below are some possible problems and items to check if you encounter them.

Problem	Check
The web interface is not responding	<ul style="list-style-type: none"> • The vovserver is down. • The vovnginxd daemon is down. • DNS is not configured to allow a route to the Monitor server host. • A firewall may be running that is blocking access to the Monitor server port (5555 by default). • If the Altair Accelerator installation or the licmon.swd is located on a network share, verify that the share can be reached from the Monitor server host.
Not receiving updated data from license servers	<ul style="list-style-type: none"> • The vovlmd daemon must be running to schedule the jobs. Check it via the Admin > Daemons page. • Check that the parser tasker is running via the Tasker page page. • Check that the load average on the system is not too high. The default parser tasker has 4 job slots and a maxload of 10.0. If the load average is too high, the parser will halt the execution of new jobs until the load average drops below the maxload value. Try to prevent other CPU-intensive processes from running on the same machine. • Check the output of the most recent parsing job via the Current > Raw Data page, or in the licmon.swd/vovlmd directory to see if the status command is having problems executing.
Alerts for a license server being monitored are intermittently thrown	<p>Status command failures are commonly seen, possibly due to network connectivity glitches, bugs in the license server status command, or other external influences. Since Monitor keeps the most recent parsing job output only by default, this log cannot be used to track down the root cause of intermittent failures. There is a advanced debug feature available to keep an archive of all parsing runs for these cases though. To enable this feature, visit the Admin > Daemons page and activate the vovtriggerd daemon. This daemon will copy (and on UNIX, compress) each parsing job log if the job fails. The copies are stored in the licmon.swd/logs/parser directory. Only use this feature when needed, since it can consume 2-3x the disk space required for normal operation.</p>
No historical data is being displayed	<ul style="list-style-type: none"> • If the installation is new, it will take at least one hour for the database to be loaded with data by default.

Problem	Check
	<ul style="list-style-type: none"> • If viewing either the Daily Statistics or Daily Plots page under the History > Features page, the summarized data for these reports are built overnight, so do not expect to see data for the current day. • Make sure that the product edition is set to Monitor and not Monitor-basic. This edition provides real-time data to the Accelerator product and does not provide historical capability. The product edition can be configured via the Admin > System page. • Verify that the license is visible and valid via the Admin > License page. • Verify that the <code>live_load_checkouts</code> task script exists in the <code>licmon.swd/tasks</code> directory. • Verify that the <code>vovdbd</code> daemon is running via the Admin > Daemons page. • Verify that the database is running via the Admin > System > Database Information page.
Drop-down menus do not stay open in Internet Explorer	This is caused by compatibility view, which forces IE to render pages and process scripts in an older engine that is not always compatible with newer web technologies. Turn compatibility view off to ensure navigation and plotting elements work correctly.
Not receiving notification e-mails	<ul style="list-style-type: none"> • The <code>vovnotifyd</code> daemon is not configured or is down. • The SMTP configuration may not be correct. Once a test mail is received, the configuration is correct and other notifications should work as well. • The <code>vovnotifyd</code> daemon uses operating system user names as the basis for email addresses. If this user name does not resolve in the email system, an email address map will need to be specified in the notification configuration.
Cannot see a specific tag anywhere in the interface	As described in General Monitoring Configuration , tags that are not actively being monitored via sampling are hidden from the user interface by default. To ensure visibility for tags that fall under these conditions, use the <code>setTagAccess</code> configuration procedure as defined in Security .

Troubleshooting: Images Do Not Display

Occasionally, images may not display correct. The images are retrieved from the readonly port (5556 by default) of the licmon server via HTTP. If your images are not displaying, try the following steps:

1. Check that readonly port is enabled.
2. Check the DNS/NIS setup and `/etc/nsswitch.conf` to be sure that the licmon hostname resolves.
3. Set the environment variable `VOV_HOST_NAME` in `licmon.swd/setup.tcl` to a value that resolves to the Monitor server machine.
4. If an IP address is the only way to access the machine from other hosts, add `VOV_HOST_HTTP_NAME` with the desired IP address to the `setup.tcl` file and perform a reread.
5. To make the change effective immediately, also enter the following commands at the shell. This sets the environment variable in the running vovserver.

```
% vovproject enable licmon
% vovsh -x "vtk_server_setenv VOV_HOST_HTTP_NAME <VALUE>"
```

If you are viewing Monitor over a port-forwarded tunnel through ssh, for example, `-L 5555:jaguar:5555`, the host names differ on each end of the connection. The only way to deal with this is to make the Monitor host an alternate name for 'localhost'. For the above example, where the remote host is 'jaguar', your line in the hosts file would be similar to:

```
127.0.0.1 localhost jaguar
```

Performance Tuning

Monitor is designed to be a highly efficient and scalable application. However, in larger enterprises, certain factors may affect the performance of the system as a whole.

Some of these factors include:

- The number of daemons being monitored
- The number of checkouts occurring across the various daemons
- Network latency
- Database size

See below for suggestions for improving the performance of the various components of Monitor.

Server

The vovserver holds the current status of all checkouts, acts as HTTP server and coordinates the execution of the parser jobs. There is no static definition of what the server requirements are. Each checkout consumes approximately 200 bytes of memory. The server requirements can roughly be calculated by estimating the number of possible concurrent checkouts that will be observed across all license servers being monitored and multiplying that number by 200 bytes. For the processor,

due to the parallel execution of the various tasks that make up Monitor's functionality, it is highly recommended to utilize a multi-core machine.

Parser Taskers

Parser taskers are responsible for running the periodic sampling jobs that collect license utilization metrics to be displayed by the server. If samples are taking a long time to finish or the time between updates is too long, there are two courses of action to consider taking. By default, there is one parser tasker that is used for sampling license servers for usage data, each with four parallel job execution slots. If the load on the machine is not too high, additional slots can be configured by adjusting the tasker's capacity setting in the `licmon.swd/taskers.tcl` file. If the load is already too high, another parser can be started on a remote machine so the workload can be distributed. To add other parser taskers, edit the `licmon.swd/taskers.tcl` file and add a code segment similar to the following:

```
# Add a parser called "parser2" on machine "jupiter"
# to distribute the load for lmstat jobs.
# This fragment goes into the licmon.swd/taskers.tcl.
vtk_tasker_define jupiter -name parser2 -executable vovtasker -rshcmd ssh -resources
  "lmparser" -maxload 10.0
  -capacity 4 -disablejobstats 1
```

After configuring an additional worker tasker, start the tasker via:

```
% vovproject enable licmon
% vovtaskermgr start
```

Starting taskers in this manner requires the ability to connect to the tasker via rsh or ssh (with password prompting disabled). For Windows, refer to the instructions for setting up vovtsd as described in [Network Monitoring](#), then specify "vovtsd" as the `-rshcmd` setting, and also pass `-vovtsdport 16666` to establish the port setting that will be used for communications.

There are also two additional taskers, one for debug log parsing jobs and one for batch reporting jobs. Additional taskers of these types can also be added if needed. Each job type is directed to an eligible tasker via its resource definition. Sampling jobs run on any tasker that offers the `lmparser` resource, as shown above in the `vtk_tasker_define` configuration procedure. Debug log parsing jobs require the `lmlogparser` resource and batch reporting jobs require `lmbatchreporter` resource. Both of these tasker types require direct access to the `licmon.swd` directory.

Data Files

Checkout Data Files

The checkout data files are generated by the various Monitor parsers, and are loaded into the database by the [vovsql_load_checkouts](#) utility. They are stored in the `licmon.swd/data/checkouts` directory. These files should always be preserved so they can be used to recreate the Monitor database in the event of a catastrophic failure. They do not need to be located in their native location once they are at least 1 day old.

The checkout data files use the following format:

- Comment lines begin with a #

- One line per checkout
- Each line is a tab separated list with the following fields:
 1. TAG
 2. NAME
 3. ACCOUNT
 4. USER
 5. HOST
 6. HANDLE
 7. DISPLAY
 8. VERSION
 9. TOKENS
 10. PID
 11. REQUEST-TIMESTAMP
 12. CHECKOUT-TIMESTAMP
 13. CHECKINTS-TIMESTAMP

Denial Data Files

The denial data files are generated by the Monitor debug log parser, and are loaded into the database by the [vovsql_load_denials](#) utility. They are stored in the `licmon.swd/data/checkouts` directory. These files should always be preserved so they can be used to recreate the Monitor database in the event of a catastrophic failure. They do not need to be located in their native location once they have been loaded into the database.

The denial data files use the following format:

- No comments
- One line per denial
- Each line is a tab separated list with the following fields:
 1. TIMESTAMP
 2. TAG
 3. FEATURE
 4. ACCOUNT
 5. USER
 6. HOST
 7. TOKENS

Capacity Data Files

The capacity data files are generated by the [ftlm_capacity](#) utility as capacity snapshots are taken (hourly by default, as the database is loaded). Only capacity changes are written. The data files are stored in the `licmon.swd/data/capacity` directory. These files should always be preserved so they can be used to recreate the Monitor database in the event of a catastrophic failure. They do not need to be located in their native location once they are at least 1 day old. The [ftlm_capacity](#) utility is also used to import these files if requested to do so.

The capacity data files use the following format:

- Comment lines begin with a #
- One line per capacity change
- Each line is a tab separated list with the following fields:
 1. setCapacity (procedure name that is called if file is imported)
 2. TAG
 3. FEATURE
 4. TOKENS
 5. TIMESTAMP

Configuration Commands

For a full list of supported license managers, see [License Server Monitoring](#).

Altair License Server

Usage

`add_ALTAIR HOST | PORT@HOST [OPTIONS]`

Options

`HOST | PORT@HOST`

If the `-cloudBased` parameter is set to false (the default), then the first argument MUST be the Altair license server host name or a combination of the port and host name, in PORT@HOST format. Otherwise this argument is ignored.

`-cmd <CMD>`

Command to run to get license utilization status.

Examples:

On premise license manager:

- `"almutil -licstat"` (use system path).
- `"/path/to/almutil -licstat"` (explicit path).

Altair cloud based license server (note that the server must be authorized separately):

- `"almutil -licstat -managed"`
- `"almutil -licstat -managed"`
- `"almutil -licstat -managed -id <LICENSE_GROUP_ID>"`

-cloudBased <TYPE>	Either false (default) or true. CMD must be consistent with the TYPE selected. If true, then the HOST PORT@HOST argument is ignored and should be set to "".
-minPeriod <PERIOD>	Minimum sampling period.
-maxPeriod <PERIOD>	Maximum sampling period.
-autokill <PERIOD>	Autokill period.
-tag <TAG>	Unique tag name. Default "ALTAIR".

Examples

```
add_ALTAIR 6200@myserver.com -tag alm_local -cmd "almutil -licstat" -cloudBased false
add_ALTAIR "" -tag alm_cloud -cmd "almutil -licstat -managed" -cloudBased true
```

Altium Log File

Usage

`add_ALTIUM_LOG </path/to/log> -tag <TAG> [OPTIONS]`

Options

<code></path/to/log></code>	The first argument must be a path to an CodeMeter™ log file. For a rotating log, the name specified must be in the form of <code><fileName>.@LATEST@.log</code> . See Debug Log Monitoring for more details.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-tz, -timezone</code>	Set to timezone which was used to generate the debug log. Ex: PST8PDT. If debug log was generated in the same timezone as the Monitor server, leave blank.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

BETA CAE License Manager

Usage

`add_BETA <HOST|PORT@HOST> [OPTIONS]`

Options

<code>HOST PORT@HOST</code>	The first argument MUST be the BETA CAE License Manager license server host name or a combination of the port and host name, in PORT@HOST format.
<code>-cmd <PATH></code>	Choose a specific sampling command. The default is 'beta_lm_stat'.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

ClearCase

Usage

`add_CLEARCASE [OPTIONS]`

Options

<code>-cmd <PATH></code>	Choose a specific sampling command. The default is 'beta_lm_stat'.
<code>-host <HOST></code>	Set artificial server host to show in reports. Default: CLEARCASE-SERVER.
<code>-port <PORT></code>	Set artificial server port to show in reports. Default: 0.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

CodeMeter Log File

Usage

```
add_CODEMETER_LOG </path/to/log> -tag <TAG> [OPTIONS]
```

Options

</path/to/log>	The first argument must be a path to an CodeMeter™ log file. For a rotating log, the name specified must be in the form of <fileName>.@LATEST@.log. See Debug Log Monitoring for more details.
-tag <TAG>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
-tz, -timezone	Set to timezone which was used to generate the debug log. Ex: PST8PDT. If debug log was generated in the same timezone as the Monitor server, leave blank.
-minPeriod <TIMESPEC>	Minimum sampling period.
-maxPeriod <TIMESPEC>	Maximum sampling period.
-autokill <TIMESPEC>	Kill parsing job after specified time.

Dassault DSLS

Usage

`add_DSLS [OPTIONS]`

Options

<code>-cmd <PATH></code>	Choose a specific sampling command. The default is 'DSLicSrv'.
<code>-host <HOST></code>	Specify license server host.
<code>-port <PORT></code>	Set artificial server port to show in reports. Default: 0.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.
<code>-usekeystore</code>	Use DSLS key store file (\$SWD/vovlmd/TAGNAME.ks) for authentication.
<code>-robust</code>	Warn and ignore unparsable license usage output lines.

DSLS Log File

Usage

```
add_DSLS_LOG </path/to/log> -tag <TAG> [OPTIONS]
```

Options

</path/to/log>	The first argument must be a path to an CodeMeter™ log file. For a rotating log, the name specified must be in the form of <fileName>.@LATEST@.log. See Debug Log Monitoring for more details.
-tag <TAG>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
-tz, -timezone	Set to timezone which was used to generate the debug log. Ex: PST8PDT. If debug log was generated in the same timezone as the Monitor server, leave blank.
-checkouts	Extract checkout information from debug log. This can be used to monitor checkouts that are shorter in duration than the sampling rate.
-host	If monitoring debug logs from triads, use this option to denote the server from which the data originates.
-minPeriod <TIMESPEC>	Minimum sampling period.
-maxPeriod <TIMESPEC>	Maximum sampling period.
-autokill <TIMESPEC>	Kill parsing job after specified time.
-usekeystore	Use DSLS key store file (\$SWD/vovlmd/TAGNAME.ks) for authentication.
-robust	Warn and ignore unparsable license usage output lines.

FlexNet Publisher

Usage

`add_LM_LICENSE_FILE <LM_LICENSE_FILE> [OPTIONS]`

Option

LM_LICENSE_FILE	The first argument MUST be the list of FlexNet Publisher servers. This may be the path to the license file or the port at host. For three-server redundant configurations, separate the list of servers with a comma (,). For multiple daemons, use colons as a separator (semi-colon for Windows).
-lmstat <PATH>	Choose a specific sampling command. The default is 'lmstat'.
-tag <TAG>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
-env <NAME>	Environment for parsing job.
-debuglog </path/to/log>	If provided, extract denial information from a debug log. For a rotating log the name specified must be in the form of <fileName>.@DATE@.log. See Debug Log Monitoring for more details. If the server being monitored is of a triad configuration or if multiple daemons are being served from the server and they each have their own debug log (meaning that their debug logs were enabled via the FlexNet Publisher options file) the <code>add_DEBUG_LOG</code> procedure must be used instead.
-debuglogcheckouts	Extract checkout information from debug log. This can be used to monitor checkouts that are shorter in duration than the sampling rate.
-site <SITE>	A short (2,3 letters typically) site specification.
-minPeriod <TIMESPEC>	Minimum sampling period.
-maxPeriod <TIMESPEC>	Maximum sampling period.
-nosplit	Keep multiple servers in one tag.
-autokill <TIMESPEC>	Kill parsing job after specified time.
-split	Split multiple servers specified by a colon-separated (semi-colon for Windows) list into individual tags (default).
-trackdisplay	The display field is normally used to indicate what display was used for a checkout. There are cases where some vendors use the display

field store information that is not related to the display and some cases where the display contains spaces and other special characters. Because of this, the display field in the checkout line is ignored by default. To override the default and enable tracking the display, use this option.



Note: It is strongly advised to verify that the display field is clean in the lmsat output for the features found in the particular vendor you are tracking before using this option.

-tracksubfeatures

A subfeature is a special field that follows the display field and is most often used to indicate that a specific function of a feature is being utilized. An example would be when a token-based feature is checked out, the subfeature will contain the tool that was enabled as a result of checking out the token-based feature. Some vendors use spaces in this field, or the display field (see above), which will result in false subfeatures. Because of this, the subfeatures field is ignored by default. To override the default and enable tracking subfeatures, use this option.



Note: It is strongly advised to verify that the subfeatures field is clean in the lmsat output for the features found in the particular vendor you are tracking before using this option.

-trackreservations

Use this option to enable tracking reservations that are displayed in the lmsat output. Reserved licenses that are not used are considered checkouts by Monitor if this option is enabled. The checkouts are for a special user, beginning with "r:", that includes the reservation object name, so that these checkouts can be distinguished from real usage in the Monitor reports. Enabling this option allows Monitor to quickly and easily identify waste that is created as a result of obsolete and overly aggressive reservations. Using this option is also required if you are intent on utilizing the "Analyze Reservation Overdraft Usage" functionality within the Detailed Plots report.

-tz <TIMEZONE>

When parsing a debug log that was generated in a different time zone, specify that timezone with this option so the times will be translated to reflect the localized time zone. Example (PST8PDT).

-minStatPeriod <TIMESPEC> Minimum sampling period for usage.

-maxStatPeriod <TIMESPEC> Maximum sampling period for usage.

-minInfoPeriod <TIMESPEC> Minimum sampling period for expiration dates.

- maxInfoPeriod <TIMESPEC> Maximum sampling period for expiration dates.
- minDebugLogPeriod
<TIMESPEC> Minimum sampling period for debug logs.
- maxDebugLogPeriod
<TIMESPEC> Maximum sampling period for debug logs.
- infoAutokill <TIMESPEC> Kill info parsing job after specified time.
- statAutokill <TIMESPEC> Kill stat parsing job after specified time.
- debugAutokill <TIMESPEC> Kill debug log parsing job after specified time.

FlexNet Publisher Debug Log

Usage

`add_DEBUG_LOG </path/to/log> -tag <TAG> [OPTIONS]`

Options

<code></path/to/log></code>	The first argument must be a path to a FlexNet Publisher debug log. For a rotating log the name specified must be in the form of <code><fileName>.@DATE@.log</code> . See Debug Log Monitoring for more details. If the server being monitored is of a triad configuration or if multiple daemons are being served from the server and they each have their own debug log (meaning that their debug logs were enabled via the FlexNet Publisher options file), each log will need its own line along with appropriate host specifications.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-tz, -timezone</code>	Set to timezone which was used to generate the debug log. Ex: PST8PDT. If debug log was generated in the same timezone as the Monitor server, leave blank.
<code>-checkouts</code>	Extract checkout information from debug log. This can be used to monitor checkouts that are shorter in duration than the sampling rate.
<code>-host</code>	If monitoring debug logs from triads, use this option to denote the server from which the data originates.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

FlexNet Publisher Options File

Usage

`add_LIC_OPTIONS </path/to/optfile> -tag <TAG> [OPTIONS]`

Options

<code></path/to/optionfile></code>	The first argument must be a path to a FlexNet Publisher options file.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

Fujitsu ICAD (Windows only)

Usage

`add_ICAD [OPTIONS]`

Options

- | | |
|--|--|
| <code>-cmd <PATH></code> | Choose a specific sampling command. The default is <code>LicMon.bat</code> , a wrapper provided by Fujitsu that ensures ASCII output. |
| <code>-tag <TAG></code> | Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (<code>_</code>). The use of upper case characters is recommended for style purposes. |
| <code>-minPeriod <TIMESPEC></code> | Minimum sampling period. |
| <code>-maxPeriod <TIMESPEC></code> | Maximum sampling period. |
| <code>-autokill <TIMESPEC></code> | Kill parsing job after specified time. |

GNS

Usage

`add_GNS [OPTIONS]`

Options

Options

- | | |
|--|--|
| <code>-cmd <PATH></code> | Choose a specific sampling command. The default is <code>licstat ALL serverhost</code> and must be configured to match the GNS license server environment. |
| <code>-tag <TAG></code> | Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (<code>_</code>). The use of upper case characters is recommended for style purposes. |
| <code>-minPeriod <TIMESPEC></code> | Minimum sampling period. |
| <code>-maxPeriod <TIMESPEC></code> | Maximum sampling period. |
| <code>-autokill <TIMESPEC></code> | Kill parsing job after specified time. |

Green Hills Legacy (Elan)

Usage

`add_GHS_LEGACY [OPTIONS]`

- | | |
|--|--|
| <code>-cmd <PATH></code> | Choose a specific sampling command. Default: <code>elmadmin</code> . |
| <code>-host <HOST></code> | Set artificial server host to show in reports. Default: <code>GHS-SERVER</code> . |
| <code>-port <PORT></code> | Set artificial server port to show in reports. Default: <code>0</code> . |
| <code>-tag <TAG></code> | Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (<code>_</code>). The use of upper case characters is recommended for style purposes. |
| <code>-minPeriod <TIMESPEC></code> | Minimum sampling period. |
| <code>-maxPeriod <TIMESPEC></code> | Maximum sampling period. |
| <code>-autokill <TIMESPEC></code> | Kill parsing job after specified time. |

LSTC

Usage

`add_LSTC [OPTIONS]`

Options

- | | |
|--|---|
| <code>-cmd <PATH></code> | Choose a specific sampling command. Default: <code>lstc_qrun</code> . |
| <code>-tag <TAG></code> | Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes. |
| <code>-minPeriod <TIMESPEC></code> | Minimum sampling period. |
| <code>-maxPeriod <TIMESPEC></code> | Maximum sampling period. |
| <code>-autokill <TIMESPEC></code> | Kill parsing job after specified time. |

LUM

Usage

add_LUM [OPTIONS]

Options

-cmd <PATH>	Choose a specific sampling command. The default is <code>LUMblt</code>
-col	Specify that the LUM server being monitored is serving concurrent offline licenses.
-host <HOST>	
-port <PORT>	Set artificial server port to show in reports. Default: 0.
-tag <TAG>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
-trackdenials	If provided, denial information from denials in addition to usage.
-tracktestlics	Track common LUM test licenses, such as iFOR Test Product and HAL Test Product.
-tz, -timezone	Set to timezone which was used to generate the debug log. Ex: PST8PDT. If debug log was generated in the same timezone as the Monitor server, leave blank.
-minStatPeriod <TIMESPEC>	Minimum sampling period for usage.
-maxStatPeriod <TIMESPEC>	Maximum sampling period for usage.
-maxInfoPeriod <TIMESPEC>	Maximum sampling period for expiration dates.
-maxInfoPeriod <TIMESPEC>	Maximum sampling period for expiration dates.
-minDenialPeriod <TIMESPEC>	Minimum sampling period for denials.
-maxDenialPeriod <TIMESPEC>	Maximum sampling period for denials.
-infoAutokill <TIMESPEC>	Kill info parsing job after specified time.
-statAutokill <TIMESPEC>	Kill stat parsing job after specified time.

-denialAutokill <TIMESPEC> Kill denial parsing job after specified time.

Wolfram MathLM

Usage

`add_MATHLM [OPTIONS]`

Options

`-cmd <PATH>` Choose a specific sampling command. The default is `monitorlm`.
(STUB)

OLicense

Usage

`add_OLICENSE [OPTIONS]`

Options

<code>-cmd <PATH></code>	Choose a specific sampling command. The default is <code>olixtool</code> .
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

QF-Test

Usage

`add_QFTEST [OPTIONS]`

Options

<code>-cmd <PATH></code>	Choose a specific sampling command. The default is <code>qftest -batch -licenseserver.dump</code> .
<code>-passwd xxxxxx</code>	Password for your license server if it is required.
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

Process Tracking

Usage

`add_PROCESS_tracking <EXECUTABLES> -tag <TAG> [OPTIONS]`

Options

EXECUTABLES	A list of the executable names to be monitored. Must be quoted if more than one in the list.
-tag TAG	A tag for the specified executable(s), for accounting purposes.
-daemon DAEMON	Artificial daemon name for reporting purposes.
-lmhost HOST	Artificial host name for reporting purposes.
-limport PORT	Maximum sampling period.
-autokill <TIMESPEC>	Kill parsing job after specified time.
-total NUMBER	Total number of licenses for each of the specified executable(s). Leave blank or specify UNLIMITED for unlimited capacity.
-sharing TYPE	Consolidate multiple process instances into one on a per user, per host, or per user/host basis.

Remote Altair Monitor

Usage

```
add_REMOTE_LM -host <HOST> -port <PORT> -site <SITE> <TAGSPEC> [OPTIONS]
```

Options

-host	Remote Altair Monitor server host.
-port	Remote Altair Monitor server port.
-minPeriod <TIMESPEC>	Minimum sampling period.
-maxPeriod <TIMESPEC>	Maximum sampling period.
-autokill <TIMESPEC>	Kill parsing job after specified time.

Tag Specification Options

-remotetag	Remote tag to monitor. Ignored if -tagprefix is passed.
-localtag	Local tag name for remote tag. Ignored if -tagprefix is passed.
-tagprefix	Monitor all remote tags and use specified prefix to create local tag name.

Examples

```
add_REMOTE_LM -host sb1m01 -port 5555 -site "SantaBarbara" -tagprefix SB_  
add_REMOTE_LM -host dfwlm01 -port 5555 -site "Dallas" -tagprefix DFW_  
add_REMOTE_LM -host nylm01 -port 5555 -site "NewYork" -remotetag MENTOR -localtag  
NY_MENTOR
```

Altair License Key File (RTDAKEY)

Usage

`add_RTDAKEY <TAG> <PORT@HOST> [OPTIONS]`

Options



- minPeriod <TIMESPEC> Minimum sampling period.
- maxPeriod <TIMESPEC> Maximum sampling period.
- autokill <TIMESPEC> Kill parsing job after specified time.

Sentinel HASP (Windows Only)

Usage

`add_HASP <HOST> [OPTIONS]`

Options


HOST	The first argument MUST be the Sentinel license server host name.
-cmd <PATH>	Choose a specific sampling command. The default is <code>vovhaspstat</code> . <div> Note: The full path to this command is <code>\$VOVDIR/../../winNT/bin/vovhaspstat.exe</code></div>
-tag <TAG>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
-trackdisplay	The display field is normally used to indicate what display was used for a checkout. There are cases where some vendors use the display field store information that is not related to the display and some cases where the display contains spaces and other special characters. Because of this, the display field in the checkout line is ignored by default. To override the default and enable tracking the display, use this option. <div> Note: It is strongly advised to verify that the display field is clean in the status output for the features found in the particular vendor you are tracking before using this option.</div>
-minPeriod <TIMESPEC>	Minimum sampling period.
-maxPeriod <TIMESPEC>	Maximum sampling period.
-autokill <TIMESPEC>	Kill parsing job after specified time.

Sentinel RMS

Usage

```
add_SENTINEL <HOST> [OPTIONS]
```

Options

HOST	The first argument MUST be the Sentinel license server host name.
-cmd <PATH>	Choose a specific sampling command. The default is <code>lsmon</code> . It is important to note that some versions of Sentinel's™ <code>lsmon</code> utility on Windows have the behavior of requiring the enter key to be pressed at the end of its output dump. This means that when executed as a background task in Monitor, the task will never end. If the <code>lsmon</code> utility being interfaced with exhibits this behavior, make a copy of the <code>vovlsmon.bat</code> script template found in <code>\$VOVDIR/bat</code> and place it somewhere outside of the <code>\$VOVDIR</code> directory. Modify it to point to the <code>lsmon.exe</code> file being interfaced with, then configure the status command to point to this script instead of <code>lsmon.exe</code> directly.
-tag <TAG>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (_). The use of upper case characters is recommended for style purposes.
-trackdisplay	The display field is normally used to indicate what display was used for a checkout. There are cases where some vendors use the display field store information that is not related to the display and some cases where the display contains spaces and other special characters. Because of this, the display field in the checkout line is ignored by default. To override the default and enable tracking the display, use this option. <div data-bbox="596 1390 630 1423"></div> Note: It is strongly advised to verify that the display field is clean in the status output for the features found in the particular vendor you are tracking before using this option.
-minPeriod <TIMESPEC>	Minimum sampling period.
-maxPeriod <TIMESPEC>	Maximum sampling period.
-autokill <TIMESPEC>	Kill parsing job after specified time.

Silvaco

Usage

`add_SILVACO [OPTIONS]`

Options

- | | |
|--|--|
| <code>-cmd <PATH></code> | Choose a specific sampling command. The default is <code>sflm</code> . |
| <code>-tag <TAG></code> | Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (<code>_</code>). The use of upper case characters is recommended for style purposes. |
| <code>-host <HOST></code> | Set artificial server host to show in reports. Default: Obtain from <code>sflm</code> output. |
| <code>-port <PORT></code> | Set artificial server port to show in reports. Default: Obtain from <code>sflm</code> output. If not present, 0. |
| <code>-minPeriod <TIMESPEC></code> | Minimum sampling period. |
| <code>-maxPeriod <TIMESPEC></code> | Maximum sampling period. |
| <code>-autokill <TIMESPEC></code> | Kill parsing job after specified time. |

T-Systems

Usage

`add_TSYSTEMS [OPTIONS]`

Options

- | | |
|--|--|
| <code>-cmd <PATH></code> | Choose a specific sampling command. The default is <code>licman20_appl</code> . |
| <code>-tag <TAG></code> | Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (<code>_</code>). The use of upper case characters is recommended for style purposes. |
| <code>-minPeriod <TIMESPEC></code> | Minimum sampling period. |
| <code>-maxPeriod <TIMESPEC></code> | Maximum sampling period. |
| <code>-autokill <TIMESPEC></code> | Kill parsing job after specified time. |

X-Formation LM-X

Usage

`add_LMX <HOST|PORT@HOST> [OPTIONS]`

Options

<code>HOST PORT@HOST</code>	The first argument MUST be the LM-X license server host name or a combination of the port and host name, in <code>PORT@HOST</code> format.
<code>-cmd <PATH></code>	Choose a specific sampling command. The default is <code>lmxendutil</code> .
<code>-tag <TAG></code>	Specify the TAG to identify the license server. The TAG may contain only alphanumeric characters and underscores (<code>_</code>). The use of upper case characters is recommended for style purposes.
<code>-minPeriod <TIMESPEC></code>	Minimum sampling period.
<code>-maxPeriod <TIMESPEC></code>	Maximum sampling period.
<code>-autokill <TIMESPEC></code>	Kill parsing job after specified time.

Autostart Directory

With the command `vovautostart`, on `vovserver` startup, scripts can be specified to execute automatically.

In UNIX, the scripts can be written in either C-shell or Tcl syntax.

 **Note:** For the script to work in Windows, Tcl syntax must be used. Guidelines follow:

- Create a directory named `autostart` in the server working directory.
- For both UNIX and Windows:
 - Create a script with the suffix `.tcl` in the `autostart` directory.
- For UNIX only, CSH scripts are also supported:
 - Create a script with the suffix `.csh` in the `autostart` directory.
 - Ensure the script has the appropriate executable permissions.

Each script in the `autostart` directory is called with one argument, which is the word `start`. This argument is usually ignored in OEM scripts, but can be used to in custom scripts to enforce different behaviors between a manual call on the CLI versus an automated call by the `vovserver`.

Examples are available in the directory `$VOVDIR/etc/autostart`.

vovautostart

The scripts are launched by the utility `vovautostart`. To repeat the execution of the `autostart` scripts, `vovautostart` can be executed from the command line.

```
vovautostart: Usage Message
```

DESCRIPTION:

Execute the scripts in the `*.swd/autostart` directory.

There are three types of scripts that get executed:

1. Scripts that match `*.sh` are executed directly (Unix only)
2. Scripts that match `*.csh` are executed directly (Unix only)
2. Scripts that match `*.tcl` are executed by `vovsh`.

The scripts are executed in alphabetical order in the background, with a 5s a 5s delay between successive scripts.

This utility is normally invoked by `vovserver` upon launching.

USAGE:

```
% vovautostart [optional directory spec]
```

EXAMPLES:

```
% vovautostart
```

Data Collection Commands

Altium Log File

```
ftlm_parse_altiumlog: Usage Message
```

SYNOPSIS:

Parses and loads checkouts found in an Altium debug log.

USAGE:

```
% ftlm_parse_altiumlog <ARGUMENTS> [OPTIONS]
```

ARGUMENTS:

- | | |
|-------------------------------|---|
| <code>-dir <dir></code> | -- Directory that contains debug logs to parse.
Only files named with the format of <code>SS.YYYYMMDD</code> are parsed. |
| <code>-f <file></code> | -- Specific log file to parse. Ignored if <code>-dir</code> is passed. On Windows, use <code>/</code> for the path delimiter and <code>//</code> before each space if there are spaces in the path. |
| <code>-h</code> | -- Show help message. |

OPTIONS:

- | | |
|---------------------------------------|--|
| <code>-tag <tag></code> | -- Tag to be used for data contained in file. |
| <code>-host <host></code> | -- Name of license server host. Default: master. |
| <code>-start <timestamp></code> | -- Ignore all events before this timestamp. |
| <code>-tz <timezone></code> | -- Set to timezone which was used to generate the debug log. Ex: <code>PST8PDT</code> . If debug log was generated in the same timezone as the LM server, leave blank. |

```
-noload          -- Do not load parsed data into the database.
-loadonly        -- Do not parse debug log for new data, only load
                  previously parsed data into the database.
-force          -- Reparse data that has been previously parsed.
-q              -- Set verbosity to 0.
-v              -- Increase verbosity (may be repeated).
```

EXAMPLES:

```
% ftlm_parse_altiumlog -tag ALTIUM -dir /remote/licenses/logs
% ftlm_parse_altiumlog -tag ALTIUM -f SS.20100701.log -tz EST5EDT
% ftlm_parse_altiumlog -tag ALTIUM -f SS.log -tz UTC-8
```

BETA CAE BETA LM

ftlm_parse_beta: Usage Message

USAGE:

```
ftlm_parse_beta [OPTIONS] TAG COMMAND
```

OPTIONS:

```
-h              -- Print help message.
-v              -- Increase output verbosity.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_beta BETA beta_lm_stat -h betalmsrv1 -a
% ftlm_parse_beta BETA beta_lm_stat -h 6007@betalmsrv1 -a
% ftlm_parse_beta BETA beta_lm_stat -h betalmsrv1,betalmsrv2,betalmsrv3 -a
```

ClearCase

ftlm_parse_clearcase: Usage Message

USAGE:

```
ftlm_parse_clearcase [OPTIONS] TAG COMMAND ARGS
```

OPTIONS:

```
-h              -- Print help message.
-v              -- Increase output verbosity.
-host           -- Specify license server host.
-port          -- Specify license server port.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLE:

```
% ftlm_parse_clearcase CLEARCASE clearlicense
% ftlm_parse_clearcase -host CLEARCASE-SERVER -port 7744
                        CLEARCASE clearlicense
```

Wibu Codemeter Log File

ftlm_parse_codemeterlog: Usage Message

SYNOPSIS:

Parses and loads checkouts found in a Code Meter debug log.

USAGE:

% ftlm_parse_codemeterlog <ARGUMENTS> [OPTIONS]

ARGUMENTS:

-dir <dir>	-- Directory that contains debug logs to parse. Only files named with the format of CodeMeter*.log are parsed.
-f <file>	-- Specific log file to parse. Ignored if -dir is passed. On Windows, use / for the path delimiter and // before each space if there are spaces in the path.
-h	-- Show help message.

OPTIONS:

-tag <tag>	-- Tag to be used for data contained in file.
-host <host>	-- Name of license server host. Default: master.
-start <timestamp>	-- Ignore all events before this timestamp.
-tz <timezone>	-- Set to timezone which was used to generate the debug log. Ex: PST8PDT. If debug log was generated in the same timezone as the LM server, leave blank.
-noload	-- Do not load parsed data into the database.
-loadonly	-- Do not parse debug log for new data, only load previously parsed data into the database.
-force	-- Reparse data that has been previously parsed.
-q	-- Set verbosity to 0.
-v	-- Increase verbosity (may be repeated).

EXAMPLES:

```
% ftlm_parse_codemeterlog -tag CodeMeter -dir /remote/licenses/logs
% ftlm_parse_codemeterlog -tag CodeMeter
                        -f CodeMeter2016-02-04-140221.log -tz EST5EDT
% ftlm_parse_codemeterlog -tag CodeMeter
                        -f CodeMeter2016-02-04-140221.log -tz UTC-8
```

Dassault DSLS

ftlm_parse_dsls: Usage Message

USAGE:

ftlm_parse_dsls [OPTIONS] TAG HOST PORT COMMAND

OPTIONS:

-h	-- Print help message.
-v	-- Increase output verbosity.
-usekeystore	-- The server is secured by a password. The parser will look for a <TAGNAME>.ks file in the vovlmd directory and use it for authentication (see DSLS docs).
-robust	-- Ignore unparsable license usage lines

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_dslogs DASSAULT dslssrv 9999 DSLicSrv
```

Dassault DSLS Log File

ftlm_parse_dslogs: Usage Message

SYNOPSIS:

Parses and loads denials, and optionally checkouts, found in a Dassault DSLS debug log.

USAGE:

```
% ftlm_parse_dslogs <ARGUMENTS> [OPTIONS]
```

ARGUMENTS:

```
-dir <dir>          -- Directory that contains debug logs to parse.
                    Only files named with the format of debug.
                    YYYYMMDD are parsed.
-f <file>           -- Specific log file to parse. Ignored if -dir
                    is passed. On Windows, use / for the path
                    delimiter and // before each space if there
                    are spaces in the path.
-h                 -- Show help message.
```

OPTIONS:

```
-tag <tag>          -- Tag to be used for data contained in file.
-host <host>        -- Name of license server host. Default: master.
-start <timestamp>  -- Ignore all events before this timestamp.
-tz <timezone>      -- Set to timezone which was used to generate the
                    debug log. Ex: PST8PDT. If debug log was
                    generated in the same timezone as the LM
                    server, leave blank.
-checkouts          -- Parse checkout data in addition to denial data.
-noload            -- Do not load parsed data into the database.
-loadonly          -- Do not parse debug log for new data, only load
                    previously parsed data into the database.
-force            -- Reparse data that has been previously parsed.
-q                -- Set verbosity to 0.
-v                -- Increase verbosity (may be repeated).
```

EXAMPLES:

```
% ftlm_parse_dslogs -tag DSLSLOG -dir /dassault/ldslogs
% ftlm_parse_dslogs -tag DSLSLOG -f LicenseServer20140630221630.log
                    -tz EST5EDT
% ftlm_parse_dslogs -tag DSLSLOG -f LicenseServer20140630221630.log
                    -tz UTC-8
```

FlexNet Publisher

ftlm_parse_flexlm: Usage Message

SYNOPSIS:

```
ftlm_parse_flexlm [OPTIONS] TAG stat|info LMSTAT_COMMAND ...
```

OPTIONS:

- v -- Increase verbosity.
- site "SITENAME" -- Assign site for source of data.
- trackdisplay -- Track display field. The display field is normally used to indicate what display was used for a checkout. There are cases where some vendors use the display field store information that is not related to the display and some cases where the display contains spaces and other special characters. Because of this, the display field in the checkout line is ignored by default.
- tracksubfeatures -- Track subfeature field. A subfeature is a special field that follows the display field and is most often used to indicate that a specific function of a feature is being utilized. An example would be when a token-based feature is checked out, the subfeature will contain the tool that was enabled as a result of checking out the token-based feature. Some vendors use spaces in this field, or the display field (see above), which will result in false subfeatures. Because of this, the subfeatures field is ignored by default.
- trackallsubfeatures -- Track all subfeatures, even those not in the list. If this is on, it implies '-tracksubfeatures' is on, and also track subfeatures not in the list, possibly false ones.
- trackreservations -- Track reservations. Reserved licenses that are not used are considered checkouts by LicenseMonitor if this option is enabled. The checkouts are for a special user, beginning with "r:", that includes the reservation object name, so that these checkouts can be distinguished from real usage in the LicenseMonitor reports. Enabling this option allows LicenseMonitor to quickly and easily identify waste that is created as a result of obsolete and overly aggressive reservations.

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_flexlm SNPS stat lmstat -a -c 1727@pluto
% ftlm_parse_flexlm -site "NewYork" SNPS stat lmstat -a -c 1727@pluto
```

FlexNet Publisher Debug Log

ftlm_parse_debuglog: Usage Message

SYNOPSIS:

Parses and loads denials, and optionally checkouts, found in a FLEXlm debug log. If TIMESTAMP lines are not included in the log file, either the -initialday argument must be used, or the debug log file name must include the starting date in the format of YYYYMMDD or YYYY.MM.DD. For parsing debug logs from triads, use the -host option to denote the server from which the data originates.

USAGE:

```
% ftlm_parse_debuglog <ARGUMENTS> [OPTIONS]

ARGUMENTS:
  -dir <dir>          -- Directory that contains debug logs to parse.
                       Only files named with the format of debug.
                       YYYYMMDD are parsed.
  -f <file>           -- Specific log file to parse. Ignored if -dir is
                       passed. On Windows, use / for the path delimiter
                       and // before each space if there are spaces in
                       the path.
  -h                  -- Show help message.

OPTIONS:
  -initialday <date>  -- Specify initial date in MM/DD/YYYY format.
                       Required if log file was created with options
                       file, or if log lacks an initial TIMESTAMP
                       statement.
  -host <host>        -- Name of license server host. If parsing debug
                       logs from triads, use this option to denote the
                       server from which the data originates.
                       Default: master.
  -tag <tag>          -- Tag to be used for data contained in file.
  -start <timestamp>  -- Ignore all events before this timestamp.
  -tz <timezone>      -- Set to timezone which was used to generate the
                       debug log. Ex: PST8PDT. If debug log was
                       generated in the same timezone as the LM
                       server, leave blank.
  -checkouts          -- Parse checkout data in addition to denial data.
  -noload             -- Do not load parsed data into the database.
  -loadonly           -- Do not parse debug log for new data, only load
                       previously parsed data into the database.
  -nomerge            -- The checkout records are loaded with origin=6,
                       which means that no merging can be performed.
                       Note: As no merging algorithm is implemented
                       for now, so this is equivalent to origin=2
                       (i.e. logs)
  -force             -- Reparse data that has been previously parsed.
  -q                 -- Set verbosity to 0.
  -v                 -- Increase verbosity (may be repeated).

EXAMPLES:
% ftlm_parse_debuglog -tag EDA -dir /remote/licenses/logs
% ftlm_parse_debuglog -tag CDN -f debug.log -initialday 07/01/2010
% ftlm_parse_debuglog -tag SNP -f debug.20100701.log -tz EST5EDT
% ftlm_parse_debuglog -tag MGC -f debug.log -checkouts -tz UTC-8
```

FlexNet Publisher Options File

```
ftlm_parse_flexlmoptions: Usage Message

DESCRIPTION:
  A parser to track the change of FLEXlm License option files.

SYNOPSIS:
  ftlm_parse_flexlmoptions [OPTIONS] TAGNAME FILE

OPTIONS:
  -v      -- Increase verbosity
  -h      -- This help
```

EXAMPLES:

```
% ftlm_parse_flexlmoptions SNPS license.opt  
% ftlm_parse_flexlmoptions -v -v CDS license2.opt
```

Fujitsu ICAD

ftlm_parse_icad: Usage Message

SYNOPSIS:

A script to parse the output of Fujitsu's ICAD license manager. This license manager is known to output unsupported 2-byte characters (namely, Japanese characters). If parsing such an instance, a wrapper script will need to be used that converts these characters to ASCII characters.

USAGE:

ftlm_parse_icad [OPTIONS] TAG COMMAND

OPTIONS:

-h -- Print help message.
-v -- Increase output verbosity.

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_icad ICAD LicMon.bat
```

GNS

ftlm_parse_gns: Usage Message

USAGE:

ftlm_parse_gns [OPTIONS] TAG COMMAND

OPTIONS:

-h -- Print help message.
-v -- Increase output verbosity.

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_gns GNS licstat ALL serverhost
```

Green Hills

ftlm_parse_ghs: Usage Message

USAGE:

ftlm_parse_ghs [OPTIONS] TAG COMMAND


```
OPTIONS:
  -h                -- Print help message.
  -v                -- Increase output verbosity.
  -acp <timespec>  -- Artificial Check-in period.

NOTE:
  On Windows, use / for the path delimiter and // before each space
  if there are spaces in the command path.

EXAMPLES:
  % ftlm_parse_ghs GHS find_ghs_licenses -w
```

Green Hills Legacy (Elan)

ftlm_parse_elan: Usage Message

```
SYNOPSIS:
  Executes the Elan license status command and parses its output to
  discover capacity and utilization information so that it can be recorded
  into server memory.
```

```
SYNTAX:
  ftlm_parse_elan TAG [OPTIONS]
```

```
OPTIONS:
  -command COMMAND  -- Specify status command.
                     Default: elmadmin -l (rely on path)
  -daemon DAEMON    -- Specify license server daemon (informational only).
                     Default: ghs
  -port PORT        -- Specify license server port (informational only).
                     Default: 0
  -site SITENAME    -- Specify license server site.
                     Default: local
  -type TYPE        -- Specify license server type.
                     Default: ghs
  -h                -- Show this help.
  -v                -- Increase output verbosity, can be repeated.
  -quiet            -- Do not echo status command output.
```

```
NOTE:
  On Windows, use / for the path delimiter and // before each space if there
  are spaces in the command path.
```

```
EXAMPLES:
  % ftlm_parse_ghs ELAN
  % ftlm_parse_ghs ELAN -site "NewYork" -command "/opt/local/bin/elmadmin"
```

LSTC

ftlm_parse_lstc: Usage Message

```
USAGE:
  ftlm_parse_lstc [OPTIONS] TAG COMMAND
```

```
OPTIONS:
```

```
-h          -- Print help message.
-v          -- Increase output verbosity.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_lstc LSTC lstc_qrun
% ftlm_parse_lstc LSTC lstc_qrun -s lstcsrv1
```

IBM LUM

ftlm_parse_lum: Usage Message

USAGE:

```
ftlm_parse_lum [OPTIONS] TAG stat|info COMMAND ARGS
```

OPTIONS:

```
-h          -- Print help message.
-v          -- Increase output verbosity.
-host <lum_server> -- Constrain to specified LUM servers. Separate
multiple servers with commas.
-port <port> -- Specify license server port for display purposes.
-tracktestlics -- Enable tracking of common test licenses found in
LUM, such as iFOR Test Product and HAL Test
Product.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_lum LUM stat i4blt -s
% ftlm_parse_lum -host lum1,lum2,lum3 LUM info i4blt -lp -i
% ftlm_parse_lum -port 7744 LUM stat ssh rmtsrv LUMblt -s
```

IBM LUM Denials

ftlm_parse_lum_denials: Usage Message

DESCRIPTION:

Parse a LUM report for denial events.

SYNOPSIS:

```
% ftlm_parse_lum_denials -tag TAGNAME [OPTIONS]
```

OPTIONS:

```
-cmd          -- Specify LUM command to use. Default: LUMblt.
-host <lum server> -- Constrain to specified LUM servers. Use commas
to separate multiple hosts.
-noload       -- Do not load denial data into database.
-tz <timezone> -- Set to timezone which was used to generate the
denial events. Ex: PST8PDT.
If denial events were generated in the same
timezone as the LM server, leave blank.
```

```
-tracktestlics      -- Enable tracking of common test licenses found in
                      LUM, such as iFOR Test Product and HAL Test
                      Product.
-h                  -- Show this help.
-v                  -- Increase verbosity. Repeatable.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_lum_denials -tag LUM1 i4blt -r 1
% ftlm_parse_lum_denials -tag LUM1 -host lnxlum1 ssh rmtsrv i4blt -r 1
% ftlm_parse_lum_denials -tag LUM1 -host lnxlum1,lnxlum2
  -tz EST5EDT i4blt -r 1
```

Wolfram MathLM

ftlm_parse_mathlm: Usage Message

USAGE:

```
ftlm_parse_mathlm [OPTIONS] TAG COMMAND
```

OPTIONS:

```
-h                  -- Print help message.
-v                  -- Increase output verbosity.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_mathlm MATHLM monitorlm
```

OLicense

ftlm_parse_olicense: Usage Message

USAGE:

```
ftlm_parse_olicense [OPTIONS] TAG COMMAND
```

OPTIONS:

```
-h                  -- Print help message.
-v                  -- Increase output verbosity.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_olicense OLicense olixtool -OLicenseServer
  simpack-srv01:17010
```

QF-Test

```
ftlm_parse_qftest: Usage Message

USAGE:
    ftlm_parse_qftest [OPTIONS] TAG COMMAND

OPTIONS:
    -h                -- Print help message.
    -v                -- Increase output verbosity.

NOTE:
    On Windows, use / for the path delimiter and // before each space if there
    are spaces in the command path.

EXAMPLES:
    % ftlm_parse_qftest QFTEST /usr/local/qftest/bin/qftest -batch
      -licenseserver.dump -licenseserver.password xxxxxxxx
```

Remote LicenseMonitor

```
lm_parse_remotelm: Usage Message

Utility that will obtain and load data from a remote LicenseMonitor instance
into a local instance.

    lm_parse_remotelm -host <host> -site <site> <TAGSPECS> [OPTIONS]

Arguments:

    -host                Define the remote LM server host. This can
                        be either a DNS name or IP address.
    -site                Define the site name that will be displayed
                        in the local LM instance.

Tag Specs:

    -tagprefix <prefix> By default, all remote tags are processed.
                        Use this argument to specify a name prefix
                        for local copy of remote tags.
    -remotetag <tag>    Specify a single remote tag from which to
                        gather data.
    -localtag <tag>     Used to specify the local tag name when
                        -remotetag is used. If specified,
                        -localtagprefix setting is ignored.

Options:

    -port <port>        Remote LM server port. Defaults to 5555.

Examples:

    % lm_parse_remote -host uklmsrv -remotetag snps -localtag UK_snps
    % lm_parse_remote -host uklmsrv -tagprefix UK_
```

Reprise RLM

```
ftlm_parse_rlmstat: Usage Message

USAGE:
    ftlm_parse_rlmstat [OPTIONS] TAG HOST PORT COMMAND

OPTIONS:
    -h                -- Print help message.
    -v                -- Increase output verbosity.
    -acp <timespec>  -- Artificial Check-in period.

NOTE:
    On Windows, use / for the path delimiter and // before each space if there
    are spaces in the command path.

EXAMPLES:
    % ftlm_parse_rlmstat RTDA localhost 7070 rlmstat -a
```

Altair License Key File (RTDAKEY)

```
ftlm_parse_rtdakey: Usage Message

DESCRIPTION:
    A parser to track the usage of VOV Key Licenses.

SYNOPSIS:
    ftlm_parse_rtdakey [OPTIONS] TAGNAME PROJECT@HOST:PORT

OPTIONS:
    -v          -- Increase verbosity
    -h          -- This help

EXAMPLES:
    % ftlm_parse_rtdakey LMKEY licmon@jupiter:5555
    % ftlm_parse_rtdakey -v -v NCKEY vnc@pluto:6271
```

Sentinel HASP

```
ftlm_parse_hasp: Usage Message

USAGE:
    ftlm_parse_hasp [OPTIONS] TAG HOST COMMAND

OPTIONS:
    -h                -- Print help message.
    -v                -- Increase output verbosity.
    -trackdisplay     -- Track display field. The display field is normally
                        used to indicate what display was used for a
                        checkout. There are cases where some vendors use
                        the display field store information that is not
                        related to the display and some cases where the
                        display contains spaces and other special
                        characters. Because of this, the display field in
```

the checkout line is ignored by default.

NOTES:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_haspl HASP sentlmsrv1 vovhaspstat
```

Sentinel RMS

ftlm_parse_sentinel: Usage Message

USAGE:

```
ftlm_parse_sentinel [OPTIONS] TAG COMMAND
```

OPTIONS:

-h	-- Print help message.
-v	-- Increase output verbosity.
-trackdisplay	-- Track display field. The display field is normally used to indicate what display was used for a checkout. There are cases where some vendors use the display field store information that is not related to the display and some cases where the display contains spaces and other special characters. Because of this, the display field in the checkout line is ignored by default.

NOTES:

Some versions of Sentinel's lsmon utility on Windows have the behavior of requiring the enter key to be pressed at the end of its output dump. This means that when executed as a background task in LicenseMonitor, the task will never end. If the lsmon utility being interfaced with exhibits this behavior, make a copy of the vovlsmon.bat script template found in \$VOVDIR/bat and place it somewhere outside of the \$VOVDIR directory. Modify it to point to the lsmon.exe being interfaced with, then configure the status command to point to this script instead of lsmon.exe directly.

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_sentinel SENTINEL lsmon sentlmsrv1
```

Silvaco

ftlm_parse_silvaco: Usage Message

USAGE:

```
ftlm_parse_silvaco [OPTIONS] TAG COMMAND
```

OPTIONS:

-h	-- Print help message.
-v	-- Increase output verbosity.
-host	-- Set artificial server host to show in reports. Default: Obtain from sflm output.

```
-port          -- Set artificial server port to show in reports.  
                Default: Obtain from sflm output.  
                If not present, 0.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_silvaco SILVACO sflm  
% ftlm_parse_silvaco SILVACO /remote/tools/silvaco/bin/sflm
```

T-Systems LICMAN

ftlm_parse_tsystems: Usage Message

USAGE:

```
ftlm_parse_licman [OPTIONS] TAG COMMAND
```

OPTIONS:

```
-h          -- Print help message.  
-v          -- Increase output verbosity.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_tsystems TSYSTEMS /opt/tsystems/bin/licman20_appl
```

X-Formation LM-X

ftlm_parse_lmx: Usage Message

USAGE:

```
ftlm_parse_lmx [OPTIONS] TAG COMMAND
```

OPTIONS:

```
-h          -- Print help message.  
-v          -- Increase output verbosity.
```

NOTE:

On Windows, use / for the path delimiter and // before each space if there are spaces in the command path.

EXAMPLES:

```
% ftlm_parse_lmx LMX lmxendutil -licstat  
% ftlm_parse_lmx LMX lmxendutil -licstat -host lmxsrv  
% ftlm_parse_lmx LMX lmxendutil -licstat -host lmxsrv -port 6200
```

Database Loading Commands

Checkouts

```
vovsql_load_checkouts: Usage Message

USAGE:
  % vovsql_load_checkouts [OPTIONS] <listOfFiles>

OPTIONS:
  -h                -- This help.
  -v                -- Increase verbosity.
  -q                -- Quiet.
  -origin <N>       -- Specify origin of data (see below)
  -f                -- Same as -force.
  -force            -- Force reloading of data files.
  -convert          -- Convert the old data files to current format.

REQUIRED ARGUMENTS:
  <listOfFiles>     -- A list of checkout data files. Can also be specified
                      as a glob expression (e.g. checkouts/2009.12.*)
                      or can contain the tokens @TODAY@ and @YESTERDAY@.

ORIGIN:
  1                -- The data comes from the sampling (default).
  2                -- The data comes from debug log parsing, to be merged.
  6                -- The data comes from debug log parsing,
                      not to be merged.

EXAMPLES:
  % vovsql_load_checkouts          licmon.swd/data/checkouts/2007*
  % vovsql_load_checkouts -force   licmon.swd/data/checkouts/2007*
  % vovsql_load_checkouts -origin 2
    licmon.swd/data/checkouts/MGC/mgcld/master/2010*
  % vovsql_load_checkouts          licmon.swd/data/checkouts/@TODAY@
```

Denials

```
vovsql_load_denials: Usage Message

USAGE:
  vovsql_load_denials [OPTIONS] <datafile> ...

OPTIONS:
  -h                -- This help.
  -v                -- Increase verbosity.
  -q                -- Quiet.
  -f, -force        -- Force reloading of data files.
  <logfile>         -- A denial data file.
                      Can be specified as a glob expression.
                      Can be repeated.

EXAMPLES:
  % cd `vovserverdir` -p data`
  % vovsql_load_denials ./denials/*/*/2008*
```


Maintenance Tasks

Maintenance tasks are managed via the Configuration Information page. Jobs for these tasks are displayed in the periodic job list page.

SNAPSHOT_CAPACITY

Take a snapshot of license capacities and loads it into the database.

Default interval is hourly.

UPDATE_SUMMARIES

Executes SQL queries against the database to gather information that is populated in summary tables. These tables are used in certain reports to give statistics more quickly than running the raw query against the database.

Default interval is nightly.

CLEANUP_SYSTEM

Executes the vovcleanup utility to perform routine cleanup of the system.

Default interval is nightly.

LOAD_REMOTE_DATA

Loads remote site data into the database.

Default interval is weekly.

UPDATE_PROJECTS

Updates project assignments as configured in the `accounts.cfg` configuration file.

Default interval is hourly.

UPDATE_MEMBERSHIPS_FROM_FILE

Updates the custom group memberships in the database based on the configuration.

Default interval is nightly.

UPDATE_MEMBERSHIPS_FROM_LDAP

Updates the custom group memberships in the database based on LDAP attributes.

Default interval is weekly.

UPDATE_MEMBERSHIPS_FROM_NIS

Updates the custom group memberships in the database based on NIS maps.

Default interval is weekly.

Accuracy

This section describes the accuracy of the Monitor sampling methodology with respect to monitoring live servers for checkout and checkin information.

Sampling Methodology

The primary sampling method is to utilize the license manager status command on a periodic basis, parsing its output to find availability and utilization information. In versions prior to 2010.06, Monitor honored the checkout time reported by the status command. Throughout many experiences in the field, it has been discovered that this can lead to inaccurate utilization data. Some of the following scenarios have been encountered at customer sites, prompting the sampling methodology to change:

1. The status command fails to return all or part of the requested information, returning cleanly. If an active checkout was expected in the status command output, but is missing due to this type of failure, a premature checkin is created and loaded into the Monitor database. If a subsequent sampling comes back with a complete dataset and the checkout is once again present, a duplicate checkout is created using the same checkout time as the one already checked in, and eventually an attempt will be made to load it into the database. In versions prior to 2010.06, the database allowed duplicate checkouts (using a unique fingerprint of feature/checkout-time/user/handle).
2. The status command reports an inaccurate checkout time altogether. This has been seen when a live checkout is removed by the license manager checkout removal utility. If and when the tool reacquires a checkout, the checkout time is expected to be the reacquisition time, but instead is something between the original checkout time (for the checkout that was removed) and the reacquisition time. This new checkout overlaps the previous checkout due to the incorrect checkout time, causing an over-reporting of usage.

These scenarios, among others, as well as the fact that the reported checkout time for FlexNet Publisher is reported with 1-minute granularity, have prompted a change in the Monitor sampling methodology. In version 2010.06 and higher, the sampling time is used for the checkout time if the checkout was not detected in a previous sampling. If the checkout is already detected, the system treats the checkout as a continuation of the pre-existing checkout. This removes the reliance upon the status command to be accurate with respect to the checkout time. The two above scenarios are now handled in the following ways:

1. For scenario 1, because the sampling time is used, there will be no duplicate checkout because the checkout times will be different. In the error condition mentioned, there will simply be a brief glitch in the continuation of the checkout that lasts as long as the error condition persists in the status command. As an additional level of protection, the database schema prevents duplicate checkouts from being loaded, based on the unique fingerprint of feature/checkout-time/user/handle.
2. For scenario 2, because the sampling time is used for the checkout time for new checkouts, the checkout time reported by the status command does not affect the checkout time in Altair Monitor, therefore, no overlap of usage occurs.

Unavailable Data

The following items are not typically available from the license manager status command:

- Process ID of the process for the checkout.
- Project (for FlexNet Publisher, the value of the LM_PROJECT variable).

- IP address of the host for the checkout.
- Checkin information. Monitor uses the absence of a previously-detected checkout as an indication of a checkin.
- Very short checkouts that are less than the sampling rate (default 30s).

Even though some short checkouts may escape detection in a sample, the long-term utilization average represents the usage with a fairly high degree of accuracy because of how the sample captures a snapshot of all utilization at each sampling time. If a higher degree of accuracy is desired, either debug log monitoring or wrapping techniques should be considered.

Access Control List

An Access Control List (ACL) is a list of permissions that are attached to an object. The list defines who can access the object (an agent) and what actions the agent can perform on the object.

Overview

The VOV software implements compartmentalized access control with Access Control Lists (ACLs). Each ACL is a triplet:

- A VOV object
- An *agent*, which is VOV security role name or an individual user name
- A *capability*, which is a controlled activity

For any user to be authorized to perform a controlled capability or action on a VOV object, an ACL must exist that contains that user or role, the controlled action, and the VOV object.

Objects

Every ACL is associated with a VOV object. Types of objects currently include:

- FairShare groups
- Resource maps
- Nodes (transitions, aka jobs, places, aka files)
- Node sets
- Reservations

Agents

Permission to perform a controlled action depends on the user ID, and the VOV role associated with that user. The `SWD/security.tcl` file defines the association of Users with Roles.

VOV's roles serve two purposes:

1. Control queue/instance/project operations (via the VOV protocol)
2. Establish high-level permissions on VOV objects

VOV has these named roles:

ADMIN

Can do just about anything. By default, the "owner" of the queue/instance/project is the only admin.

LEADER	Can do lots of things, but not everything an ADMIN can.
USER	Can create and manage their own objects.
READONLY	Can view most things, but not create.
ANYBODY	Very limited, mainly used for testing.
NOBODY	Nothing, mainly used for testing.
UNKNOWN	The last resort, in case we somehow encounter someone who doesn't fall into any of the above roles, also can do nothing.

An ACL is expressed in terms of operations that are permitted to an *agent* acting on the object. An agent may be a USER (login account), an OS group (OSGROUP), a FairShare group (FSGROUP), a machine (HOST) or one of the symbolic agents EVERYBODY, OWNER, ADMIN. The most powerful agent is the SERVER.

ACLs support the following "agents", which provide the identities of the persons involved:

USER <i>name</i>	OS authorized user
OWNER - role	The queue/instance/project owner
USER - role	The user that owns the VOV object
OSGROUP <i>name</i>	Members of the specified OS group
FSGROUP <i>name</i>	Members of the specified FairShare group
HOST	Anyone with a client connected from a specific host
EVERYBODY - role	Everybody
ADMIN - role	Anyone with the ADMIN role
SERVER	The vovserver process, specifically
LEADER	Anyone with the LEADER role
USERGROUP <i>name</i>	Members of the specified VOV user group
UNDEF	The last resort, in case there is someone who doesn't fall into any of the above agent types.

For the agents that are groups, membership in the group confers the operations permitted by that ACL. For example, if the login `joe` is a member of the OS group `dvregr`, and OSGROUP `dvregr` has APPEND on a `fsgroup`, then `joe` may add ACLs to that `fsgroup`.

To bypass the ACL, you must be the logged in on the host running vovserver as the user that is running vovserver, and you must change VOV_HOST_NAME to "localhost".

ACL Management

To perform ACL management, use a utility with the following syntax:

```
% vovacl [OPTIONS] <Objects>
```

The following utilities are available for ACL management:

Utility	Description
vovacl	Script to manage ACLs in VOV.

ACL Commands

ACL management consists of the following commands:


Command	Description
APPEND	Add ACLs to an object.
DELETE	Delete an ACL element from an object. The element is identified by the agent and name fields.
GET	Get current ACLs on an object. It shows you the current ACLs that are associated with an object, if the ACL permits you to VIEW it.
RESET	Reset ACLs on an object to defaults. It removes all the object's current ACLs and replaces them with the default values. <div>ACL 1: OWNER "" ATTACH DETACH EDIT VIEW FORGET DELEGATE EXISTS ACL 2: EVERYBODY "" ATTACH VIEW</div>

ACL Actions

Following are the actions that can be controlled via ACLs:

Action	Description
ATTACH	Create a relationship between objects
CHOWN	Change ownership of an object
CREATE	Create an object

Action	Description
DELEGATE	Assign ACLs on an object
DETACH	Destroy a relationship between objects
EDIT	Modify properties of an object
EXIST	The agent is aware of the existence of the object
FORGET	Forget an object
RESUME	Resume a suspended job
SIGNAL	Send a signal to a job.
STOP	Stop an object.
SUSPEND	Suspend an object
VIEW	View properties of an object

 **Note:** Not all actions apply to all objects. In the case of FairShare groups, applicable actions include: ATTACH, EDIT, VIEW, DELEGATE. The actions RETRACE, STOP, SUSPEND, FORGET are reserved for use with jobs in future releases.


Obtain SERVER Credentials

For some ACL operations, you will need the most powerful credentials, that is, SERVER, which are only available to the owner of the vovserver process when connected on the loopback interface.

- Login on the vovserver host as the user that is running vovserver.
- Enable the project with `vovproject enable PROJECTNAME`.
- Change the VOV_HOST_NAME to `localhost`

```
% setenv VOV_HOST_NAME localhost
```

- Now your clients act as the SERVER agent with respect to the ACL.

 **Note:** The "OSGROUP name" ACLs are not relevant when using the web browser interface. This is because web browsers provide no OS group information to the servers processing the requests.

Widgets

A Monitor "widget" is a web page that generates various reports, but with no interface or visual aids. Widgets are called by other pages that provide these items, as well as details about the content provided by the widget, such as a title and description of the content.

Widgets are useful in the following scenarios:

- When custom report pages or page formats are desired.
- When it is desired to have Monitor reports embedded into other websites. The language used by the website doesn't matter, as long as it has the capability of including content from another URL (such as HTML's inline frame). For an example (using PHP) of a page with embedded widgets, see the Widget Application Example.

Using Widgets

Widgets are accessed through a URL via Monitor's [read-only port](#). The URL parameters that are passed to the widget define its behavior.

The root URL for all widgets is:

```
http://lm-host:lm-readonly-port/cgi/ftlm.cgi?widget=widget-name&m1=value1
```

By supplying a widget name at the end of the URL, along with the required parameters for the widget, a report will be generated with the UI and some visual elements absent. A good way to learn how to use widgets is to experiment with a widget in its own browser page. This can be accomplished by clicking on the popout icon in the top right-hand corner of an existing Monitor report. Most reports in Monitor are driven by widgets. Clicking on the help icon in this area opens this documentation page.

After opening a widget in its own window, change the parameters in the browser's address bar and note how the widget updates when the URL is loaded.

Widget Summary Listing

Here is a quick listing of all widgets available in the Monitor system. Click on the highlighted name for further details

Output Type	Name	Description
Tabular	checkouts	View of current checkouts.
Tabular	checkoutstats	History checkout statistics.
Tabular	checkoutdets	Detailed history of checkouts.
Tabular	tags	Report on current server information.
Tabular	tagstats	
Tabular	features	Current feature usage.

Output Type	Name	Description
Tabular	featurestats	Feature usage statistics.
Tabular	featureeffstats	Feature usage efficiency.
Tabular	denialstats	Report of denial statistics.
Tabular	denialdets	Detailed view of denials.
Graphical	checkouthistogram	Distribution of historical checkouts.
Graphical	featurestatsplot	Visualization of feature usage.
Graphical	featureeffhistogram	Visualization of feature efficiency.
Graphical	featuredetsplot	Detailed graph and pie charts.
Graphical	denialplot	Visualize denials over time.
Graphical	heatmap	Visualize feature usage over a week cycle.
Graphical	piechart	Get a pie chart of usage for each user.

Detailed Widget Listing

The following section describes the available widgets and provides example URLs and descriptions of the options for each widget:

Current Checkouts

Widget Name checkouts

Example URL <http://host:port/cgi/ftlm.cgi?widget=checkouts>

Widget URL options:

Parameter	Default	Possible Values	Meaning
reportby	taggedfeature	tag, feature, taggedfeature, user, host, account	Report by what.

Parameter	Default	Possible Values	Meaning
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
user		any user name	Show report on this user only.
host		any host name	Show report on this host only.
account		any account name	Show report on this account only.
version		any version number	Show report on this version only.
filter_tag		any tag name pattern	Filter on tag name. Only show the checkouts with tags matching the pattern.
filter_feature		any feature name pattern	Filter on feature name. Only show the checkouts with name matching the pattern.
filter_user		any user name pattern	Filter on user name. Only show the checkouts with tags matching the pattern.
filter_host		any host name pattern	Filter on host name. Only show the checkouts with tags matching the pattern.
filter_account		any account name pattern	Filter on account name. Only show the checkouts with tags matching the pattern.
filter_version		any version pattern	Filter on version. Only show the checkouts with versions matching the pattern.

Parameter	Default	Possible Values	Meaning
limit	30	any positive integer	How many checkouts to show on the page.
ascend	0	0 or 1	Sort in ascending (ascend=1) or descending (ascend=0) order.
sortby	handles	tag, feature, taggedfeature, user, host, account, count	Sort on which column.
nocase	0	0 or 1	Ignore case for user and host.

Checkout Statistics

Widget Name: checkoutstats

Example URL <http://host:port/cgi/ftlm.cgi?widget=checkoutstats&timerange=Yesterday>

Widget URL options:

Parameter	Default	Possible Values	Meaning
reportby	taggedfeature	tag, feature, taggedfeature, user, host, account	Report by what.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
user		any user name	Show report on this user only.
host		any host name	Show report on this host only.
account		any account name	Show report on this account only.

Parameter	Default	Possible Values	Meaning
filter_tag		custom tag name string or regex	Report on only those tags that match above string or regex.
filter_feature		custom feature name string or regex	Report on only those features that match above string or regex.
filter_user		custom user name string or regex	Report on only those users that match above string or regex.
filter_host		custom host name string or regex	Report on only those hosts that match above string or regex.
filter_account		custom account name string or regex	Report on only those accounts (projects) that match above string or regex.
filter_version		custom version name string or regex	Report on only those versions that match above string or regex.
limit	100	integer number	Max number of records to show.
sortby	handles	tag, feature, taggedfeature, user, host, account, count, dur_tot, dur_avg, dur_max, wait_count, wait_tot, wait_avg, wait_max	Sort on which column.
ascend	0	0 or 1	Sort in ascending (ascend=1) or descending (ascend=0) order.
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start/end options are present.

Parameter	Default	Possible Values	Meaning
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.

Checkout Details

Widget Name: checkoutdets

Example URL <http://host:port/cgi/ftlm.cgi?widget=checkoutdets>

Widget URL options:

Parameter	Default	Possible Values	Meaning
sortby	checkouttime	duration, checkouttime, checkintime, queuedtime, tag, feature, user, host, version, handle, origin, tokens, project, reservations, location	Sort checkouts by what.
reportby	taggedfeature	tag, feature, taggedfeature, user, host, account	Report by what.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
user		any user name	Show report on this user only.
host		any host name	Show report on this host only.
account		any account (project) name	Show report on this account only.

Parameter	Default	Possible Values	Meaning
version		any version name	Show report on this version only.
filter_tag		custom tag name string or regex	Report on only those tags that match above string or regex.
filter_feature		custom feature name string or regex	Report on only those features that match above string or regex.
filter_user		custom user name string or regex	Report on only those users that match above string or regex.
filter_host		custom host name string or regex	Report on only those hosts that match above string or regex.
filter_account		custom account name string or regex	Report on only those accounts (projects) that match above string or regex.
filter_version		custom version name string or regex	Report on only those versions that match above string or regex.
limit	100	integer number	Max number of rows to show.
sortby	handles	tag, feature, taggedfeature, user, host, account, count, dur_tot, dur_avg, dur_max, wait_count, wait_tot, wait_avg, wait_max	Sort on which column.
ascend	0	0 or 1	Sort in ascending (ascend=1) or descending (ascend=0) order.
timerange		Yesterday, Last Week, This Week, Last Month,	Time period on which to report. Ignored when

Parameter	Default	Possible Values	Meaning
		This Month, Last Quarter, Last Year	start/end options are present.
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.

Checkout Duration Histogram

Widget Name: checkouthistogram

Example URL <http://host:port/cgi/ftlm.cgi?widget=checkouthistogram>

Widget URL options:

Parameter	Default	Possible Values	Meaning
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
tag		any tag name	Tag name of the feature on which to report.
feature		any feature name	Name of the feature on which to report.
user		any user name	Name of the user on which to report.
host		any host name	Name of the host on which to report.
account		any account name	Name of the account(project) on which to report.
version		any version number	Version on which to report.

Parameter	Default	Possible Values	Meaning
filter_tag		custom tag name string or regex	Report on only those tags that match above string or regex.
filter_feature		custom feature name string or regex	Report on only those features that match above string or regex.
filter_user		custom user name string or regex	Report on only those users that match above string or regex.
filter_host		custom host name string or regex	Report on only those hosts that match above string or regex.
filter_account		custom account name string or regex	Report on only those accounts (projects) that match above string or regex.
filter_version		custom version name string or regex	Report on only those versions that match above string or regex.
origin		samples, logs, merged	Origin of checkout data.

Current Feature Information

Widget Name: features

Example URL http://host:port/cgi/ftlm.cgi?widget=features&sortby=inuse&ascend=0&filter_tag=CDN

Widget URL options:

Parameter	Default	Possible Values	Meaning
sortby	inuse	tag, vendor, feature expire total users inuse percent oldest	Sort on which column.
ascend	0	0 or 1	Sort in ascending (ascend=1) or descending (ascend=0) order.

Parameter	Default	Possible Values	Meaning
limit	30	any positive integer	How many features to show on the page.
showfeature	active	active all perpetual expiring expired	Which features to show on the page based on various states.
tag		any tag name	Tag name of the feature on which to report.
vendor		any vendor name	Name of the vendor on which to report.
feature		any feature name	Name of the feature on which to report.
filter_tag		any tag name pattern	Filter on tag name. Only show the features with tags matching the pattern.
filter_vendor		any license daemon vendor name	Filter on vendor name. Only show the features with vendor name matching the pattern.
filter_feature		any feature name pattern	Filter on feature name. Only show the features with name matching the pattern.

Feature Statistics

Widget Name:

featurestats

Example URL

<http://host:port/cgi/ftlm.cgi?widget=featurestats>

Widget URL options:

Parameter	Default	Possible Values	Meaning
sortby	inuse	inuse, tag, feature	Sort on which column.
ascend	0	0 or 1	Sort in ascending (ascend=1) or

Parameter	Default	Possible Values	Meaning
			descending (ascend=0) order.
limit	100	any positive integer	How many features to show on the page.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
filter_tag		any tag name pattern	Filter on tag name. Only show the features with tags matching the pattern.
filter_feature		any feature name pattern	Filter on feature name. Only show the features with name matching the pattern.
showpeakavg		0 or 1	Show peak average feature usage.
origin		samples, logs, merged	Origin of checkout data.

Feature Efficiency Statistics

Widget Name: featureeffstats

Example URL <http://host:port/cgi/ftlm.cgi?widget=featureeffstats>

Widget URL options:

Parameter	Default	Possible Values	Meaning
sortby	inuse	inuse, tag, feature	Sort on which column.
ascend	0	0 or 1	Sort in ascending (ascend=1) or descending (ascend=0) order.
limit	100	any positive integer	How many features to show on the page.

Parameter	Default	Possible Values	Meaning
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
filter_tag		any tag name pattern	Filter on tag name. Only show the features with tags matching the pattern.
filter_feature		any feature name pattern	Filter on feature name. Only show the features with name matching the pattern.
showpeakavg		0 or 1	Show peak average feature usage.

Feature Efficiency

Widget Name:

featureeffhistogram

Example URL

<http://host:port/cgi/ftlm.cgi?widget=featureeffhistogram>

Widget URL options:

Parameter	Default	Possible Values	Meaning
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start= and end= are given. Case-insensitive, e.g. 'last week' means the same as 'Last Week' Older versions used the key showactive for this parameter. Since some possible values contain spaces, you need to URL-encode them, for example timerange=Last+Month

Parameter	Default	Possible Values	Meaning
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
filter_tag		any tag name pattern	Filter on tag name. Only show the features with tags matching the pattern.
filter_feature		any feature name pattern	Filter on feature name. Only show the features with name matching the pattern.
showres	0	0, 1	Show reservations (0=false, 1=true)
workweek	0	0 , 1	Limit data to the work week.
origin		samples, logs, merged	Origin of checkout data.

Feature Statistics Plot

Widget Name: featurestatsplot

Example URL <http://host:port/cgi/ftlm.cgi?widget=featurestatsplot>

Widget URL options:

Parameter	Default	Possible Values	Meaning
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start= and end= are given. Case-insensitive,

Parameter	Default	Possible Values	Meaning
			e.g. 'last week' means the same as 'Last Week' Older versions used the key showactive for this parameter. Since some possible values contain spaces, you need to URL-encode them, for example timerange=Last+Month
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
filter_tag		any tag name pattern	Filter on tag name. Only show the features with tags matching the pattern.
filter_feature		any feature name pattern	Filter on feature name. Only show the features with name matching the pattern.
plotusage	1	0 or 1	Show feature usage graphically (0=false, 1=true)
plotqueuedrequests	1	0 or 1	Show queued requests graphically (0=false, 1=true)
plotdenials	1	0 or 1	Show denials graphically (0=false, 1=true)

Feature Details Plot

Widget Name: featuredetsplot

Example URL `http://host:port/cgi/ftlm.cgi?
widget=featuredetsplot&tag=CDN&feature=NC-
Verilog&timerange=Yesterday`

Widget URL options:

Parameter	Default	Possible Values	Meaning
tag		any tag name	Tag name of the feature on which to report.
feature		any feature name	Name of the feature on which to report.
width	600	integer number	The width of the graph.
height	200	integer number	The height of the graph.
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start= and end= are given. Case-insensitive, e.g. 'last week' means the same as 'Last Week' Older versions used the key showactive for this parameter. Since some possible values contain spaces, you need to URL-encode them, for example timerange=Last+Month
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
showres	0	0, 1	Show reservations (0=false, 1=true)

Parameter	Default	Possible Values	Meaning
workweek	0	0 , 1	Limit data to the work week.

Denial Statistics

Widget Name: denialstats

Example URL <http://host:port/cgi/ftlm.cgi?widget=denialstats>

Widget URL options:

Parameter	Default	Possible Values	Meaning
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start= and end= are given. Case-insensitive, e.g. 'last week' means the same as 'Last Week' Older versions used the key showactive for this parameter. Since some possible values contain spaces, you need to URL-encode them, for example timerange=Last+Month
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
reportby	taggedfeature	tag, feature, taggedfeature, user, host, account	Report by what.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.

Parameter	Default	Possible Values	Meaning
user		any user name	Show report on this user only.
host		any host name	Show report on this host only.
reason			
filter_tag		custom tag name string or regex	Report on only those tags that match above string or regex.
filter_feature		custom feature name string or regex	Report on only those features that match above string or regex.
filter_user		custom user name string or regex	Report on only those users that match above string or regex.
filter_host		custom host name string or regex	Report on only those hosts that match above string or regex.

Denial Details

Widget Name:

denialdets

Example URL

<http://host:port/cgi/ftlm.cgi?widget=denialdets>

Widget URL options:

Parameter	Default	Possible Values	Meaning
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start= and end= are given. Case-insensitive, e.g. 'last week' means the same as 'Last Week' Older versions used the key showactive for this parameter. Since some possible values contain spaces, you need to URL-encode

Parameter	Default	Possible Values	Meaning
			them, for example timerange=Last+Month
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
tag		any tag name	Show report on this tag only.
feature		any feature name	Show report on this feature only.
user		any user name	Show report on this user only.
host		any host name	Show report on this host only.
reason	unknown	unknown, "Licensed number of users already reached", "Feature has expired", "Users are queued for this feature"	Denial reason given by debugLog.
filter_tag		custom tag name string or regex	Report on only those tags that match above string or regex.
filter_feature		custom feature name string or regex	Report on only those features that match above string or regex.
filter_user		custom user name string or regex	Report on only those users that match above string or regex.
filter_host		custom host name string or regex	Report on only those hosts that match above string or regex.
limit		integer 1 or greater	Display only the indicated number of records.

Parameter	Default	Possible Values	Meaning
querylimit		integer 1 or greater	Return only up to the number given, from the database.
sortby	handle	handle, user, tag, feature, host, reason	Sort report rows by the indicated column.
ascend	0	0 or 1	Sort in ascending order (ascend=1) or in descending order (ascend=0).

Denial Plot

Widget Name:

denialplot

Example URL

<http://host:port/cgi/ftlm.cgi?widget=denialplot>

Widget URL options:

Parameter	Default	Possible Values	Meaning
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start= and end= are given. Case-insensitive, e.g. 'last week' means the same as 'Last Week' Older versions used the key showactive for this parameter. Since some possible values contain spaces, you need to URL-encode them, for example timerange=Last+Month

Parameter	Default	Possible Values	Meaning
tag		any tag name	Tag name of the feature on which to report.
feature		any feature name	Name of the feature on which to report.
user		any user name	Show report on this user only.
host		any host name	Show report on this host only.
reason	unknown	unknown, "Licensed number of users already reached", "Feature has expired", "Users are queued for this feature"	Denial reason given by debugLog.
width	600	integer number	The width of the graph.
height	200	integer number	The height of the graph.
ymax		integer 0 or greater	Maximum Y axis value.
workweek	0	0 , 1	Limit data to the work week.

Feature Activity Heatmap

Widget Name: heatmap

Example URL <http://host:port/cgi/ftlm.cgi?widget=heatmap&type=checkout&tag=CDN&feature=NC-Verilog&timerange=Yesterday>

Widget URL options:

Parameter	Default	Possible Values	Meaning
tag		any tag name	Tag name to map.
feature		any feature name	Name of the feature on to map.

Parameter	Default	Possible Values	Meaning
type	checkout	checkout, checkin, denials	What kind of activity to report on with this heatmap.
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start/end options are present.
start		unix timestamp	Start time (in unix timestamp format) of the reporting period.
end		unix timestamp	End time (in unix timestamp format) of the reporting period.

Usage and Denial Pie Chart

Widget Name: piechart

Example URL <http://host:port/cgi/ftlm.cgi?widget=piechart&type=duration&tag=CDN&feature=NC-Verilog&timerange=Yesterday>

Widget URL options:

Parameter	Default	Possible Values	Meaning
tag		any tag name	Tag name of the feature to plot.
feature		any feature name	Name of the feature to plott.
type	count	count, duration, denials	Get piechart on what for each user.
timerange		Yesterday, Last Week, This Week, Last Month, This Month, Last Quarter, Last Year	Time period on which to report. Ignored when start/end options are present.
start		unix timestamp	Start time (in unix timestamp format) of the plot period.

Parameter	Default	Possible Values	Meaning
end		unix timestamp	End time (in unix timestamp format) of the plot period.

Current Server Information

Widget Name: tags

Example URL `http://host:port/cgi/ftlm.cgi?widget=tags&sortby=handles&ascend=0&filter_tag=MGC`

Widget URL options:

Parameter	Default	Possible Values	Meaning
sortby	handles	tag, type, vendor, licserver, status, features, handles or update	Sort on which column.
ascend	0	0 or 1	Sort in ascending (ascend=1) or descending (ascend=0) order.
limit	30	any positive integer	How many daemons to show on the page.
filter_tag		any tag name pattern	Filter on tag name. Only show the daemons with tags matching the pattern.
filter_type		flexlm, lum, etc	Filter on License Manager type, for example specify "flexlm" to show only FlexNet Publisher type license daemons.
filter_vendor		any license daemon vendor name	Filter on vendor name. Only show the daemons with vendor name matching the pattern.

Parameter	Default	Possible Values	Meaning
filter_status		status of the daemon	Filter on license server status. Only show the daemons with license server status matching the pattern.

Tag Statistics

Widget Name: tagstats

Example URL `http://host:port/cgi/ftlm.cgi?widget=tagstats`

Widget URL options:

Parameter	Default	Possible Values	Meaning
tag		Any valid tag name.	Display tag statistics for the period selected.
filter_tag		any tag name pattern	Filter on tag name. Only show the daemons with tags matching the pattern.

Widget PHP Integration Example

For this section, please refer to the example PHP file located in `$VOVDIR/training/lm/lm_widgets.php`. This example shows how to utilize Altair Monitor widgets from within a PHP page.

The example assumes that Altair Monitor is running on host "dragon" with a read-only port of 5556. Through the use of the PHP `fopen` and `fpass thru` functions, pages are generated that include a license usage graph, pie chart, and more, on the same page.

Please refer to [Widgets](#) for information on how to use each individual widget.

Time Zones

Example time zone specifications are:

UTC-8	Shanghai, Singapore. Do not use CST (China Standard Time), which is ambiguous.
UTC+05:30	India Standard Time. Do not use IST, which is ambiguous (India Standard Time or Israel Standard Time or Ireland Standard Time)
MET	Middle Europe Time, Berlin, Rome, Paris (this includes the summer time MEST)
EST5EDT	Eastern Time, New York, Atlanta
CST6CDT	Central Time, Texas. Do not use CST (Central Standard Time), which is ambiguous.
MST7MDT	Mountain Time, Colorado
PST8PDT	Pacific Time, San Francisco, Los Angeles

To find out the valid time zone names, look in one of the following locations:

```
/usr/share/zoneinfo  
/usr/share/lib/zoneinfo  
/usr/lib/zoneinfo  
/usr/local/etc/zoneinfo  
C:/Progra~1/cygwin/usr/local/etc/zoneinfo
```

Legal Notices

Intellectual Property Rights Notice

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

Copyright ©1986-2025 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of the intellectual property rights held by Altair Engineering Inc. or its affiliates. Software, other products, and materials of Altair Engineering Inc. or its affiliates are protected under laws of the United States and laws of other jurisdictions.

In addition to intellectual property rights indicated herein, such software, other products, and materials of Altair Engineering Inc. or its affiliates may be further protected by patents, additional copyrights, additional trademarks, trade secrets, and additional other intellectual property rights. For avoidance of doubt, copyright notice does not imply publication. Copyrights in the below are held by Altair Engineering Inc. or its affiliates. Additionally, all non-Altair marks are the property of their respective owners. If you have any questions regarding trademarks or registrations, please contact marketing and legal.

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

Altair HyperWorks®, a Design & Simulation Platform

Altair® AcuSolve® ©1997-2025

Altair® Activate® ©1989-2025

Altair® Automated Reporting Director™ ©2008-2022

Altair® Battery Damage Identifier™ ©2019-2025

Altair® CFD™ ©1990-2025

Altair Compose® ©2007-2025

Altair® ConnectMe™ ©2014-2025

Altair® DesignAI™ ©2022-2025

Altair® DSim® ©2024-2025

Altair® DSim® Cloud ©2024-2025

Altair® DSim® Cloud CLI ©2024-2025

Altair® DSim® Studio ©2024-2025

Altair® EDEM™ ©2005-2025

Altair® EEvision™ ©2018-2025

Altair® ElectroFlo™ ©1992-2025
Altair Embed® ©1989-2025
Altair Embed® SE ©1989-2025
Altair Embed®/Digital Power Designer ©2012-2025
Altair Embed®/eDrives ©2012-2025
Altair Embed® Viewer ©1996-2025
Altair® e-Motor Director™ ©2019-2025
Altair® ESAComp® ©1992-2025
Altair® expertAI™ ©2020-2025
Altair® Feko® ©1999-2025
Altair® FlightStream® ©2017-2025
Altair® Flow Simulator™ ©2016-2025
Altair® Flux® ©1983-2025
Altair® FluxMotor® ©2017-2025
Altair® GateVision PRO™ ©2002-2025
Altair® Geomechanics Director™ ©2011-2022
Altair® HyperCrash® ©2001-2023
Altair® HyperGraph® ©1995-2025
Altair® HyperLife® ©1990-2025
Altair® HyperMesh® ©1990-2025
Altair® HyperMesh® CFD ©1990-2025
Altair® HyperMesh® NVH ©1990-2025
Altair® HyperSpice™ ©2017-2025
Altair® HyperStudy® ©1999-2025
Altair® HyperView® ©1999-2025
Altair® HyperView Player® ©2022-2025
Altair® HyperWorks® ©1990-2025
Altair® HyperWorks® Design Explorer ©1990-2025
Altair® HyperXtrude® ©1999-2025
Altair® Impact Simulation Director™ ©2010-2022
Altair® Inspire™ ©2009-2025
Altair® Inspire™ Cast ©2011-2025
Altair® Inspire™ Extrude Metal ©1996-2025

Altair® Inspire™ Extrude Polymer ©1996-2025
Altair® Inspire™ Form ©1998-2025
Altair® Inspire™ Mold ©2009-2025
Altair® Inspire™ PolyFoam ©2009-2025
Altair® Inspire™ Print3D ©2021-2025
Altair® Inspire™ Render ©1993-2025
Altair® Inspire™ Studio ©1993-20245
Altair® Material Data Center™ ©2019-2025
Altair® Material Modeler™ ©2019-2025
Altair® Model Mesher Director™ ©2010-2025
Altair® MotionSolve® ©2002-2025
Altair® MotionView® ©1993-2025
Altair® Multi-Disciplinary Optimization Director™ ©2012-2025
Altair® Multiscale Designer® ©2011-2025
Altair® newFASANT™©2010-2020
Altair® nanoFluidX® ©2013-2025
Altair® NLView™ ©2018-2025
Altair® NVH Director™ ©2010-2025
Altair® NVH Full Vehicle™ ©2022-2025
Altair® NVH Standard™ ©2022-2025
Altair® OmniV™©2015-2025
Altair® OptiStruct® ©1996-2025
Altair® PhysicsAI™©2021-2025
Altair® PollEx™ ©2003-2025
Altair® PollEx™ for ECAD ©2003-2025
Altair® PSIM™ ©1994-2025
Altair® Pulse™ ©2020-2025
Altair® Radioss® ©1986-2025
Altair® romAI™ ©2022-2025
Altair® RTLvision PRO™ ©2002-2025
Altair® S-CALC™ ©1995-2025
Altair® S-CONCRETE™ ©1995-2025
Altair® S-FRAME® ©1995-2025

Altair® S-FOUNDATION™ ©1995-2025
Altair® S-LINE™ ©1995-2025
Altair® S-PAD™ © 1995-2025
Altair® S-STEEL™ ©1995-2025
Altair® S-TIMBER™ ©1995-2025
Altair® S-VIEW™ ©1995-2025
Altair® SEAM® ©1985-2025
Altair® shapeAI™©2021-2025
Altair® signalAI™©2020-2025
Altair® Silicon Debug Tools™©2018-2025
Altair® SimLab® ©2004-2025
Altair® SimLab® ST ©2019-2025
Altair® SimSolid® ©2015-2025
Altair® SpiceVision PRO™ ©2002-2025
Altair® Squeak and Rattle Director™ ©2012-2025
Altair® StarVision PRO™ ©2002-2025
Altair® Structural Office™ ©2022-2025
Altair® Sulis™©2018-2025
Altair®Twin Activate®©1989-2025
Altair® UDE™ ©2015-2025
Altair® ultraFluidX® ©2010-2025
Altair® Virtual Gauge Director™ ©2012-2025
Altair® Virtual Wind Tunnel™ ©2012-2025
Altair® Weight Analytics™ ©2013-2022
Altair® Weld Certification Director™ ©2014-2025
Altair® WinProp™ ©2000-2025
Altair® WRAP™ ©1998-2025

Altair HPCWorks®, a HPC & Cloud Platform
Altair® Allocator™ ©1995-2025
Altair® Access™ ©2008-2025
Altair® Accelerator™ ©1995-2025
Altair® Accelerator™ Plus ©1995-2025
Altair® Breeze™ ©2022-2025

Altair® Cassini™ ©2015-2025
Altair® Control™ ©2008-2025
Altair® Desktop Software Usage Analytics™ (DSUA) ©2022-2025
Altair® FlowTracer™ ©1995-2025
Altair® Grid Engine® ©2001, 2011-2025
Altair® InsightPro™ ©2023-2025
Altair® InsightPro™ for License Analytics ©2023-2025
Altair® Hero™ ©1995-2025
Altair® Liquid Scheduling™ ©2023-2025
Altair® Mistral™ ©2022-2025
Altair® Monitor™ ©1995-2025
Altair® NavOps® ©2022-2025
Altair® PBS Professional® ©1994-2025
Altair® PBS Works™ ©2022-2025
Altair® Simulation Cloud Suite (SCS) ©2024-2025
Altair® Software Asset Optimization (SAO) ©2007-2025
Altair® Unlimited™ ©2022-2025
Altair® Unlimited Data Analytics Appliance™ ©2022-2025
Altair® Unlimited Virtual Appliance™ ©2022-2025

Altair RapidMiner®, a Data Analytics & AI Platform
Altair® AI Hub ©2023-2025
Altair® AI Edge™ ©2023-2025
Altair® AI Cloud ©2022-2025
Altair® AI Studio ©2023-2025
Altair® Analytics Workbench™ ©2002-2025
Altair® Graph Lakehouse™ ©2013-2025
Altair® Graph Studio™ ©2007-2025
Altair® Knowledge Hub™ ©2017-2025
Altair® Knowledge Studio® ©1994-2025
Altair® Knowledge Studio® for Apache Spark ©1994-2025
Altair® Knowledge Seeker™ ©1994-2025
Altair® IoT Studio™ ©2002-2025
Altair® Monarch® ©1996-2025

Altair® Monarch® Classic ©1996-2025

Altair® Monarch® Complete™ ©1996-2025

Altair® Monarch® Data Prep Studio ©2015-2025 Altair® Monarch Server™ ©1996-2025

Altair® Panopticon™ ©2004-2025

Altair® Panopticon™ BI ©2011-2025

Altair® SLC™ ©2002-2025

Altair® SLC Hub™ ©2002-2025

Altair® SmartWorks™ ©2002-2025

Altair® RapidMiner® ©2001-2025

Altair One® ©1994-2025

Altair® CoPilot™ ©2023-2025

Altair® Drive™ ©2023-2025

Altair® License Utility™ ©2010-2025

Altair® TheaRender® ©2010-2025

OpenMatrix™ ©2007-2025

OpenPBS® ©1994-2025

OpenRadioss™ ©1986-2025

Third Party Software Licenses

For a complete list of Altair Accelerator Third Party Software Licenses, please click [here](#).

Technical Support

Altair provides comprehensive software support via web FAQs, tutorials, training classes, telephone and e-mail.

Altair One Customer Portal

Altair One (<https://altairone.com/>) is Altair's customer portal giving you access to product downloads, Knowledge Base and customer support. We strongly recommend that all users create an Altair One account and use it as their primary means of requesting technical support.

Once your customer portal account is set up, you can directly get to your support page via this link: www.altair.com/customer-support/.

Altair Training Classes

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

Index

Special Characters

.cshrc [9](#)
.profile [9](#)

A

access control list [187](#)
access to help [16](#)
accuracy [186](#)
ACL [187](#)
add_ALTNUM_LOG [142](#)
add_BETA [143](#)
add_CLEARCASE [144](#)
add_CODEMETER_LOG [145](#)
add_DEBUG_LOG [151](#)
add_DSLS [146](#)
add_DSLS_LOG [147](#)
add_GHS_LEGACY [155](#)
add_GNS [154](#)
add_HASP [165](#)
add_ICAD [153](#)
add_LIC_OPTIONS [152](#)
add_LM_LICENSE_FILE [148](#)
add_LM_LICENSE_FILE configuration file [61](#)
add_LMX [169](#)
add_LSTC [156](#)
add_LUM [157](#)
add_MATHLM [159](#)
add_OLICENSE [160](#)
add_PROCESS_tracking [162](#)
add_QFTEST [161](#)
add_REMOTE_LM [163](#)
add_RTDAKEY [164](#)
add_SENTINEL [166](#)
add_SILVACO [167](#)
add_TSYSTEMS [168](#)
adjust global defaults [50](#)
adjust parser settings [50](#)
agent configuration [95](#)
agent groups, ACL [187](#)
agent list [95](#)
agent operations, ACL [187](#)
Altair License Key File (RTDAKEY) [164](#)
altair license server command [140](#)
Altium log file [142](#)

- assign site [43](#)
- automatic backups [42](#)
- automatic data loading [38](#)
- automatically set the project for live checkouts [68](#)
- automatically start Monitor at system startup - UNIX [22](#)
- automatically start Monitor at system startup - Windows [22](#)
- autostart directory [169](#)

B

- BETA CAE License Manager [143](#)
- bin/bash [9](#)
- bin/csh [9](#)
- bin/tcsh [9](#)
- binary wrapping [85](#)
- browser-based configuration [54](#)
- bypass ACL instructions [187](#)

C

- capacity data files [138](#)
- case sensitivity [50](#)
- check and clear tag modification history [43](#)
- checkout data files [138](#)
- checkouts [40](#)
- CLEANUP_SYSTEM [185](#)
- ClearCase [144](#)
- CodeMeter log file [145](#)
- command line operations [119](#)
- command line setup [96](#)
- command versions [54](#)
- components, monitor [5](#)
- configuration [32](#), [34](#), [34](#)
- configuration commands [140](#)
- configure a failover server replacement [25](#)
- configure database from the command line [34](#)
- configure Monitor [54](#)
- configure the tls/ssl protocol [109](#)
- configuring monitoring [50](#)
- configuring user groups via the vovusergroup utility [123](#)
- control access to tag data [104](#)
- control which license usage data is loaded [38](#)
- ControlCenter, introduction [89](#)
- crash recovery mode [28](#)
- crash recovery restart [28](#)
- creating a license [85](#)
- custom group configuration activation [68](#)
- custom groups driven by FlexNet Publisher options file [68](#)

customize actions needed to enable access to Altair Accelerator products on Windows [14](#)

D

- daemon control [133](#)
- daemon manager utility [30](#)
- daemon operation [29](#)
- daemon, automatic start [169](#)
- Dassault DSLS [146](#)
- data collection commands [170](#)
- data files [138](#)
- database [32](#)
- database backup [42](#)
- database configuration options [36](#)
- database control [32](#)
- database control options [36](#)
- database daemon [32](#)
- database engine versions and upgrades [37](#), [38](#)
- database export [40](#)
- database loading commands [184](#)
- database location [32](#)
- database schema [49](#)
- database tasker [32](#)
- database tasks [32](#), [36](#)
- debug log [58](#)
 - configuration [62](#)
 - enable FlexNet Publisher [58](#)
 - file-based configuration [61](#)
 - joint monitoring [61](#)
 - rotating [58](#)
 - specification [62](#)
 - stand-alone monitoring [62](#)
 - static [58](#)
 - time zones [62](#)
 - web-based configuration [61](#)
- debugging [98](#)
- denial data files [138](#)
- denials [40](#)
- deobfuscation [46](#)
- detailed widget listing [192](#)
- determine the current engine version [37](#)
- DSLS log file [147](#)
- dump previous release databases [40](#)

E

- email map page [98](#)
- enable Altair Accelerator for non-interactive shells [10](#)

- enable CLI access on UNIX [9](#)
- enable CLI access on Windows [13](#)
- enable FlexNet Publisher debug logs [58](#)
- enable the command prompt to communicate with a running product server [15](#)
- enable the shell to communicate with a running product server [11](#)
- ensure visibility for tags not actively being monitored [50](#)
- export data [40](#)
- exported jobs directory [40](#)
- external webserver [108](#)

F

- failover configuration, tips [26](#)
- failover server candidates [25](#)
- failover.sh [26](#)
- false denial filtering [65](#)
- feature and tag specific configurations [98](#)
- file-based configuration [54](#), [78](#), [84](#), [98](#)
- first startup [21](#)
- FlexNet Publisher [148](#)
- FlexNet Publisher debug log [151](#)
- FlexNet Publisher option file [152](#)
- FlexNet Publisher reservations [54](#)
- ftlm_capacity [121](#)
- ftlm_licmgr_init [96](#)
- ftlm_parse_altiumlog [170](#)
- ftlm_parse_beta [170](#)
- ftlm_parse_clearcase [170](#)
- ftlm_parse_codemeterlog [170](#)
- ftlm_parse_debuglog [170](#)
- ftlm_parse_dsIs [170](#)
- ftlm_parse_elan [170](#)
- ftlm_parse_ftlm [170](#)
- ftlm_parse_ftlmlic [50](#)
- ftlm_parse_ftlmoptions [170](#)
- ftlm_parse_ghs [170](#)
- ftlm_parse_gns [170](#)
- ftlm_parse_hasp [170](#)
- ftlm_parse_icad [170](#)
- ftlm_parse_lmx [170](#)
- ftlm_parse_lstc [170](#)
- ftlm_parse_lum [170](#)
- ftlm_parse_lum_denials [170](#)
- ftlm_parse_mathlm [170](#)
- ftlm_parse_olicense [170](#)
- ftlm_parse_qftest [170](#)
- ftlm_parse_rlmstat [170](#)

- ftlm_parse_rtdakey [170](#)
- ftlm_parse_sentinel [170](#)
- ftlm_parse_silvaco [170](#)
- ftlm_parse_tsystems [170](#)
- ftlm_summary [122](#)
- Fujitsu ICAD [153](#)

G

- general monitoring configuration [50](#)
- get started with ControlCenter [89](#)
- GNS [154](#)
- Green Hills Legacy (Elan) [155](#)
- guest access port [107](#)

H

- help, Accelerator [16](#)
- how vovserver failover works [26](#)
- HTTP access models [107](#)

I

- images don't display [23](#), [137](#)
- instructions, bypass ACL [187](#)
- interfaces and security [5](#)
- internal webserver [108](#)

J

- joint debug log monitoring [61](#)
- journal files [28](#)

L

- LDAP authentication [102](#)
- LDAP integration [114](#)
- legacy webserver [108](#)
- license management overview [89](#)
- license managers, remote [54](#)
- license managers, supported [54](#)
- license server management [89](#)
- license server monitoring [54](#)
- license violation effects and actions [46](#)
- LicManager [96](#)
- live monitoring and manual parsing [62](#)
- lm_parse_remotelm [170](#)
- Immgr [18](#), [18](#)
- Immgr reset [24](#)

- Immgr stop [24, 28](#)
- load the database [120](#)
- LOAD_REMOTE_DATA [185](#)
- locked licenses [50](#)
- LSTC [156](#)
- LUM [157](#)

M

- maintenance tasks [185](#)
- manage data from different origins [46](#)
- manage tags in database [43](#)
- manage tags in server memory [43](#)
- manual backups [42](#)
- manual data loading [38](#)
- manually annotate the database with project information [68](#)
- method 1: use windows explorer to set command line environment [13](#)
- method 2: using Windows command prompt to set command line environment [13](#)
- Monitor overview [4](#)
- monitor processes [84](#)
- Monitor, editions [5](#)
- Monitor, features [5](#)
- Monitor, licensing [5](#)
- Monitor, theory of operation [5](#)
- monitory system diagram [5](#)
- multi-site support [66](#)

N

- ncmgr stop [28](#)
- network monitoring [78](#)
- notification [98](#)

O

- obtain server credentials [187](#)
- OLicense [160](#)
- online help [16](#)
- OS-based authentication on UNIX [102](#)
- OS-based authentication on Windows [102](#)
- other configurations [98](#)

P

- parse a debug log [120](#)
- parser taskers [137](#)
- parser.cfg [50](#)
- PDF, access [16](#)
- performance tuning [137](#)

process tracking [162](#)
project and group management [68](#)

Q

QF-Test [161](#)
query the vovserver [125](#)
quick start guide [18](#)

R

rebuild a database from source data files [43](#)
reference topics [133](#)
register security changes [103](#)
Remote Monitor [163](#)
remotely controlled monitoring agent configuration [78](#)
rename tags [43](#)
reprise, status commands [54](#)
reset Monitor [24](#)
restart, crash recovery [28](#)
restore a database from backups [42](#)
rotating debug log [58](#)

S

sample a license server [119](#)
sampling methodology [186](#)
script wrapping example [85](#)
security [102](#)
security analysis [105](#)
security principles [103](#)
Sentinel HASP [165](#)
Sentinel RMS [166](#)
server candidates [25](#)
server operation [18](#)
servercandidates.tcl [25](#), [26](#)
serverinfo.tcl [26](#)
set feature capacity [121](#)
setting up SSH keys [78](#)
setup.tcl [26](#)
Silvaco [167](#)
smtp configuration page [98](#)
SNAPSHOT_CAPACITY [185](#)
source data backups [42](#)
stand-alone debug log monitoring [62](#)
stand-alone monitoring agent configuration [78](#)
start and configure the server [30](#)
start Monitor manually on UNIX [18](#)

- start Monitor manually on Windows [20](#)
- static debug log [58](#)
- status command specification [54](#)
- stop Monitor [24](#)
- supported license managers [54](#)
- system description [5](#)

T

- T-Systems [168](#)
- tag access, configure [104](#)
- tag administration [43](#)
- tag names [54](#)
- theory of operation, Monitor [5](#)
- time zones [213](#)
- troubleshooting [133](#)
- troubleshooting images display [23](#), [137](#)
- troubleshooting the UNIX setup [12](#)

U

- unavailable data [186](#)
- update summary table [122](#)
- UPDATE_MEMBERSHIPS_FROM_FILE [185](#)
- UPDATE_MEMBERSHIPS_FROM_LDAP [185](#)
- UPDATE_MEMBERSHIPS_FROM_NIS [185](#)
- UPDATE_PROJECTS [185](#)
- UPDATE_SUMMARIES [185](#)
- upgrade the database engine [37](#)
- usage information in FlexNet Publisher debug logs [62](#)
- use Accelerator help [16](#)
- use vovselect for querying [128](#)
- user management [102](#)
- using a license [85](#)
- using widgets [191](#)

V

- verify access to Altair Accelerator products [10](#)
- verify context is working [14](#)
- view and edit ACLs [105](#)
- VOV_HOST_NAME [187](#)
- VOV_STDOUT_SPEC [16](#)
- vovautostart [25](#), [169](#)
- vovbrowser [16](#)
- vovbuild [16](#)
- vovdaemonmgr [30](#), [30](#), [133](#)
- vovdb [32](#)

- vovdb_util [34, 40, 42](#)
- vovdb_util clearcfg [36](#)
- vovdb_util configure [36](#)
- vovdb_util showcfg [36](#)
- vovdb_util startdb [36](#)
- vovdb_util upgrade [37](#)
- vovdbd [32](#)
- vovdbd tasker [32](#)
- vovdoc [16, 16](#)
- vovfsgroup [187](#)
- vovid [16](#)
- vovldap_update_memberships [68](#)
- vovlmd [43, 54, 62, 85](#)
- vovnis_update_memberships [68](#)
- vovproject [26, 34, 187](#)
- vovrc [18](#)
- vovretraced [28](#)
- vovselect [125](#)
- vovserver [16, 137, 187](#)
- vovserver autostart [169](#)
- vovserverdir [25](#)
- vovservsel.tcl [26](#)
- vovsql_load_checkouts [38, 138, 184](#)
- vovsql_load_denials [38, 138, 184](#)
- vovtasker [26](#)
- vovtaskers [25](#)
- vovtriggerd [29](#)
- vovtsd [22, 78](#)
- vovusergroups [123](#)
- vovusergroups support [123](#)

W

- web interface [111](#)
- web server configuration [107](#)
- web-based configuration [78, 84, 98](#)
- web-based custom group configuration [68](#)
- web-based project configuration [68](#)
- widget PHP integration example [213](#)
- widgets [191](#)
- widgets summary listing [191](#)
- windows domain accounts [103](#)
- Wolfram MathLM [159](#)
- working with remote license servers [54](#)
- workweek definition [118](#)
- wrapping unlicensed tools [85](#)

X

X-Formation LM-X [169](#)