ALTAIR

ONLY FORWARD

Altair Hero 2025.1.2

User Guide

Contents

Overview 1

Hero is Altair's solution for effective use of emulation resources.

This chapter covers the following:

Hero is based on Altair Accelerator and leverages its capabilities including resource management, FairShare, reservations and report generation. Hero employs an abstraction model that allows it to dispatch and manage emulators from multiple vendors even within the same queue.

The overall focus of Hero is to improve emulator resource utilization over time. While this document focuses on running emulation jobs, Hero can also effectively manage the intensive design compilation activities associated with emulation and can be used to construct a complete emulation flow, if required. It can also be used in conjunction with conventional batch systems for ordinary computing tasks.

Some example files are provided in the `$VOVDIR/training/demos/hero` directory. These files will be referenced in the Setup section.

The document describes typical usages; more details can be found in the usage messages of the individual commands or, for more general information, in the Altair Accelerator User Guide documentation.

Note: When the Hero installation is updated or patched, the installation `policy.tcl` file retains its previous values. It may be useful to compare the existing `policy.tcl` file with any updates to the file `$VOVDIR/etc/ProjectTypes/hero/policy.tcl` and make updates or remove as appropriate. The following vovserver configuration parameters may be relevant:

- maxResMap
- maxBufferSize
- resmap.max.map.length

Environment Variable Conventions

For the processes described in this document, it is assumed that the NC_QUEUE environment variable is set to the word `hero`. In practice, the queue name can be set to something meaningful to the user (some restrictions apply).

Setup

Hero is straightforward to start and configure. The steps are:

1. Create/start the Hero project.
2. Create the emulator configuration file `emul.cfg`.
3. Run the configuration command.

To start a project (and create it the first time around), use the following command:

```
hemgr start
```

This will create a project which is named `hero` by default. When using the `nc` commands, the project name must be supplied (either via `-q` or the NC_QUEUE environment variable).

The emulator configuration file needs to be set up. The configuration describes the details of the emulators managed by Hero. The file is stored in the `hero` project directory (called server working directory, SWD) which may be found using the command:

```
nc cmd vovserverdir -p
```

As an example, it may be helpful to copy the demo files `demo_data/` and `emul.cfg` found under `$VOVDIR/training/demos/hero/` into the directory `SWD/config/hero/`. The `emul.cfg` gives an example configuration file, the `demo_data/` directory contains some example files used by the example `emul.cfg` to describe the emulators.

The configuration file is named `emul.cfg` and it has a section that describes the emulators listed under EMULATORS. Most of the entries in the emulator description apply to all emulators, some are emulator specific.

- NAME is the name used to refer to the emulator in the interface. The name must be alphanumeric and be unique among emulator names.
- TYPE must be one of three types, "PalladiumZ1" (which also applies to the PZ2), "Zebu" or "HAPS".
- GROUP is a label for a group of emulators that are equivalent in the sense that a job can potentially run on any emulator in the group. For example, a number of ZeBu Z4s could be grouped together. The group name must be alphanumeric and unique among group names.
- RTHOSTS is a list of hosts on which jobs associated with this emulator will run. They must be configured by Hero so that emulator scheduling works properly.

The following entries are emulator dependent:

- TEST_SERVER_JSON is a JSON format file for the Palladium emulators that describes the emulator configuration and contains the output of the `test_server -json` command for that emulator.
- ZSE is a Tcl format file describing the layout of the ZeBu emulator. This file is part of the Synopsys emulator installation setup.

The following entries are optional:

- IGNORE_LIST is a list of emulator elements that are not to be included when determining resources where a job may run. The format is emulator dependent, the Palladium IGNORE_LIST is a list of boards, the Zebu IGNORE_LIST is a list of either unit or module names.
- Example usage would be if a module is out of service, or a cluster that is not physically present. An example for the Palladium would be `IGNORE_LIST = ["7"]` which will not schedule a job on Board 7. An example for ZeBu would be `IGNORE_LIST = ["U1", "U3.M2"]` which will not schedule a job on Unit 1 or Unit 3, Module 2.
- TASKER_ENV is a space separated list of environment variables and values that will be assigned when the taskers associated with this emulator are started (see the RTHOSTS entry). An example usage is to set the ZEBU_SYSTEM_DIR variable to identify a ZeBu emulator.

To process the configuration and start the taskers associated with the emulators, use the command:

```
nc cmd vovemulmgr config
```

It typically takes a few iterations to get the configuration file syntax correct. As an additional check, run the following command (which should print no error messages).

```
nc cmd vovemulmgr check
```

The web interface Status page provides a useful visual check.

Run and Stop a Job

An emulator job is submitted in the same manner as a typical Altair Accelerator job, with a few additional features to deal with the complexity of scheduling jobs on an emulator. The two main differences are the emulation specific resources that must be specified and the presence of a `hero_adapter` wrapper.

Submit an Emulation Job

To submit an emulation job use the following format:

```
nc run -r HERO:<group>_<target> -- hero_adapter <user command>
```

The resource `HERO:<group>_<target>` needs some elaboration. Each type of emulator defines certain targets (for example, SPAN3 corresponds to a 3 board Palladium job). The resource `HERO:<group>_<target>` conjoins all of the emulator targets in a group that define a given target into a single resource string.

As an illustration, in the demo example, to submit a two board job to the Palladium group, a two module job to the ZeBu and a HAPS job, use the following commands (the group names here are particular to the demo example):

```
nc run -r HERO:PZ1_SPAN2 -- hero_adapter sleep 20
nc run -r HERO:Z4_SPAN2 -- hero_adapter sleep 20
nc run -r HERO:HAPS100_SPAN1 -- hero_adapter sleep 20
```

The following command shows the status and placement (as appropriate) for Hero jobs:

```
nc cmd vovselect id, status, statusnc, lasttaskername, prop.HERO_EMUL_NAME,
 prop.HERO_PPLACEMENT from jobs -where resources~HERO
```

Stop an Emulation Job

Stopping an emulation job requires more care than other jobs so that the user command has an appropriate chance to respond to the signals. It is important to send the relevant signals to the appropriate process. In particular, the `vw` and `hero_adapter` wrapper commands need to be excluded.

The format of the command is as follows:

```
nc stop -skiptop 1 -exclude vovsh <jobid...>
```

The `-skiptop 1` and `-exclude vovsh` options ensure the stopping signals are passed through to the user command. For more advanced options for the `stop` subcommand please refer to the Accelerator documentation. It is likely that the signal sequence will need to be specified along with delays so that the emulator jobs can be stopped gracefully.

Some EDA tools start child processes that are not terminated when the main tool process ends which can result in unnecessary process load. The `tasker.childProcessCleanup` configuration parameter (see *Clean up Processes Left Behind by Completed Jobs*) can be used to address this issue.

Interface to Emulation Jobs

When an emulation job runs, the `hero_adapter` wrapper sets the values of some environment variables so that the emulation job can perform appropriate actions. These variables are in addition to those normally set by Accelerator when a job runs, such as VOV_JOBID.

> **Important:** The user command must use the values of these variables to set up an appropriate emulation environment for the job.
>
> An example would be to create a `.bp` placement file in the case of a Palladium emulator.

The following environment variables are set by `hero_adapter` when the job runs:

- HERO_EMUL (Example value: "z4cluster0")
- HERO_PLACEMENT (Example value: "30 31")
- HERO_PPLACEMENT (Example value: "U7.M2 U7.M3")

For the ZeBu emulators, an additional environment variable is set:

- ZEBU_PHYSICAL_LOCATION (Example value: "U6.M0,U7.M0")

The user command can use these values to determine what action to take.

In addition, each tasker associated with an emulator may have some environment variables specified in addition (see the TASKER_ENV option as described in Setup). For example, the ZEBU_SYSTEM_DIR variable is needed by ZeBu jobs.

Job Placements for ZeBu Emulators

By default, running `vovemulmgr config` creates resource maps that allow an emulation job to run on any valid Zebu placement.

On large Zebu emulators (for example, one with 16 units), there are many valid placements, and configuration may take a long time to complete, or can possibly cause the vovserver to run out of memory.

This is only an issue for jobs that require more than one unit.

These issues can be mitigated by reducing the number of legal placements generated by specifying the CONNECTIVTY entry in the relevant emulator specific section of the `emul.cfg` configuration file. The CONNECTIVTY option can take the following values which represent a trade off between configuration time and allowable placements:

- **combinations** (default): Generate all valid Zebu placements.
- **extended**: Generate a subset of the placements generated using the "combinations" value using a grouping heuristic to reduce the number of placements generated. This has the most impact for jobs that require approximately half of the number of emulator units.

- **explicit <comma separated list of targets>**: Generate placements for the specified targets. For example, `CONNECTIVITY = "explicit:SPAN1,SPAN25"` will only generate placements for the SPAN1 and SPAN25 targets.
- **sequential**: Generate placements only where the placement units are sequential.

Web Interface

In addition to the Accelerator web interface, Hero has two additional pages. The **Home** page gives a summary of the performance metrics of the emulators.

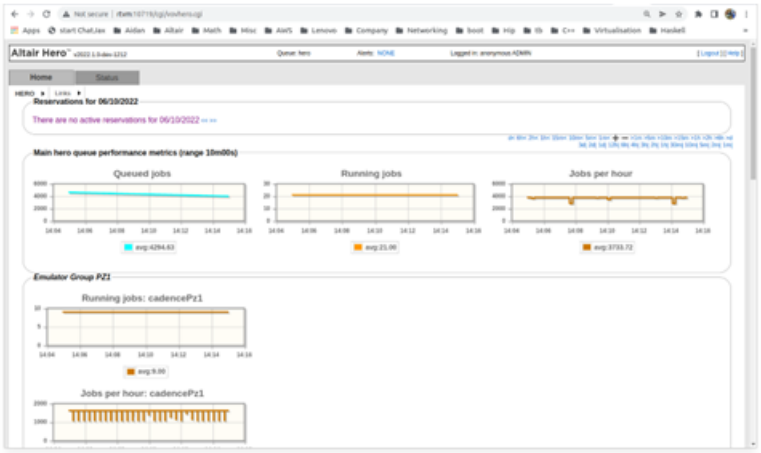

Figure 1:

The **Status** page gives a visual indication of emulator usage, the execution hosts and the jobs running on each emulator.

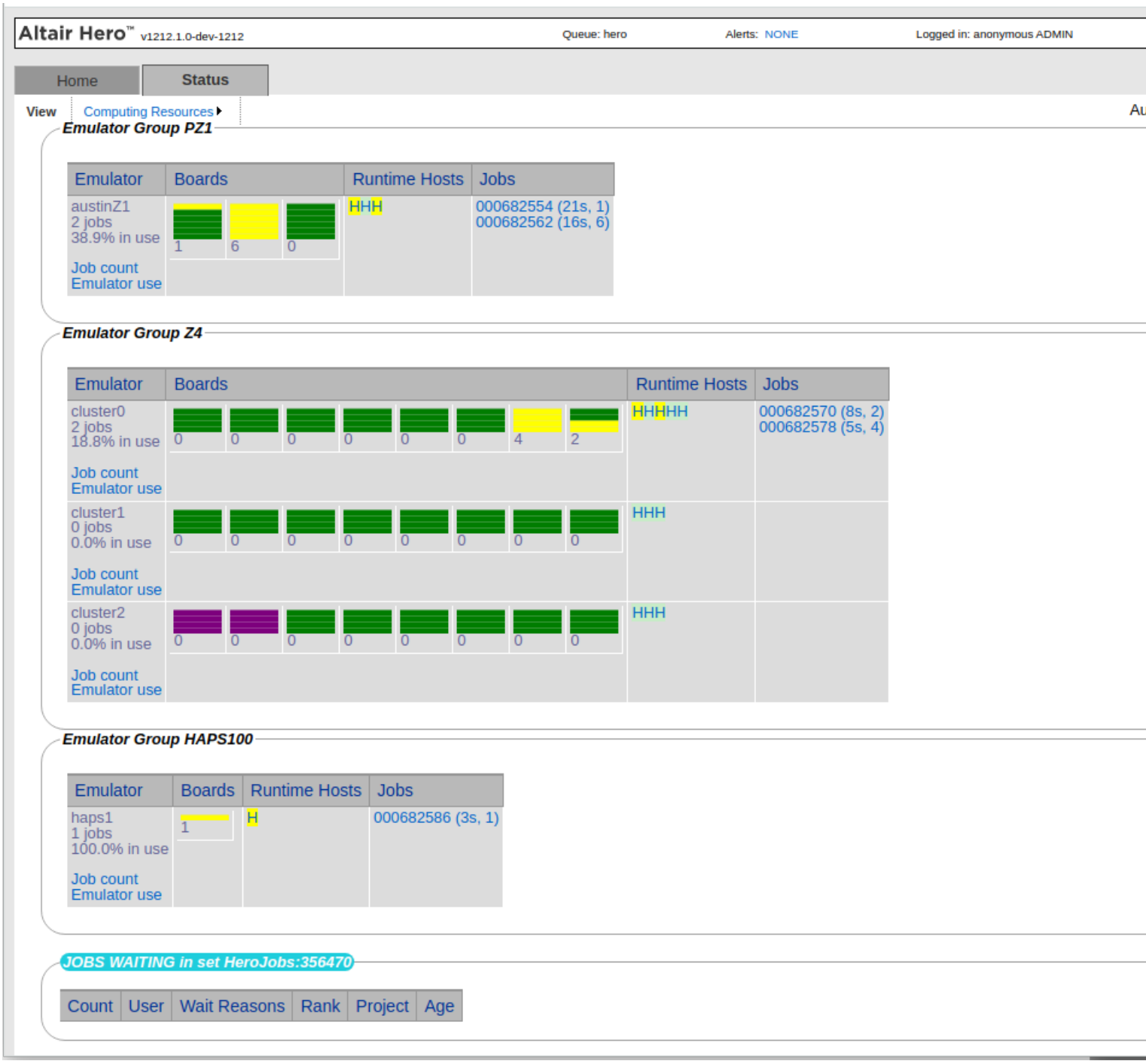

Figure 2:

The web interface can be accessed using the URL returned by the following command:

```
nc cmd vovbrowser
```

Reservations

Reservations can be used to reserve an emulator or a portion of an emulator for a specified interval of time and for a particular class of users such as user, FairShare group, job ID, job class or job project.

The `hero_reserve` command provides a convenient interface for making emulator reservations. The syntax of the command is given below and varies slightly depending on the type of emulator. The general format is as follows:

```
nc cmd hero_reserve HERO:<emulator-name>_<leaf-resource-list-spec> [who the
 reservation is for] [interval of time]
```

For example, to reserve boards 0 through 5 on a Palladium emulator named cadencePz1 for an hour for job project gpu_integ, the following command could be used:

```
nc cmd hero_reserve 'HERO:LEAF_cadencePz1_(0..5)' -jobproj gpu_integ -duration 1h
```

Similarly, to reserve all modules on units 6 and 7 on a ZeBu emulator named z4cluster0 the following command could be used:

```
nc cmd hero_reserve 'HERO:LEAF_z4cluster0_U(6..7).M(0..3)' -jobproj gpu_integ -
duration 1h
```

The reservations can be listed using the command:

```
nc cmd vovshow -w -reservations
```

A reservation can be removed using the following command:

```
nc cmd vovforget <reservation id>
```

(The reservation id can be obtained from the `vovshow` command.)

In the above examples, only a job submitted with job project `gpu_integ` can start on the corresponding portion of the emulator.

The `hero_reserve` command has an additional feature that creates a reservation based on the target resource, for example:

```
nc cmd hero_reserve 'HERO:PZ1_SPAN2' -jobproj gpu_integ -duration 1h
```

However, this method does not allow the emulator or board range to be specified.

Use Altair Monitor for Reporting

To enable tracking of emulator usage, Hero acts as a license server in which each license or token corresponds to an element of the emulator in use. When a job runs, the appropriate number of emulator tokens are checked out and returned when the job finishes.

For example, in the earlier example, if the job with request HERO:PZ1_SPAN2 starts running on the emulator named cadencePz1, 16 (2 boards of 8 domains each) licenses of the license Emul:cadencePz1 will be checked out and returned when the job terminates. This unusual representation of emulator resources allows Altair Monitor to be used to report both real time and historical emulator usage.

Altair Monitor has many reporting options (see Altair Accelerator Administrator's Guide, but only the most basic setup is described here as an illustration. The setup requires the port and host of the Hero project which can be obtained with the command:

```
nc cmd vovselect port,host from server
```

One other thing to note is that the command `vlmstat` is used to probe the Hero server and is analogous to `lmstat`. Go to Monitor's **Admin** tab and add a new license monitor using the host, port and vlmstat command. You should choose a tag name that can represent the Hero queue.

When jobs run, usage can be tracked from the License Manager web interface:

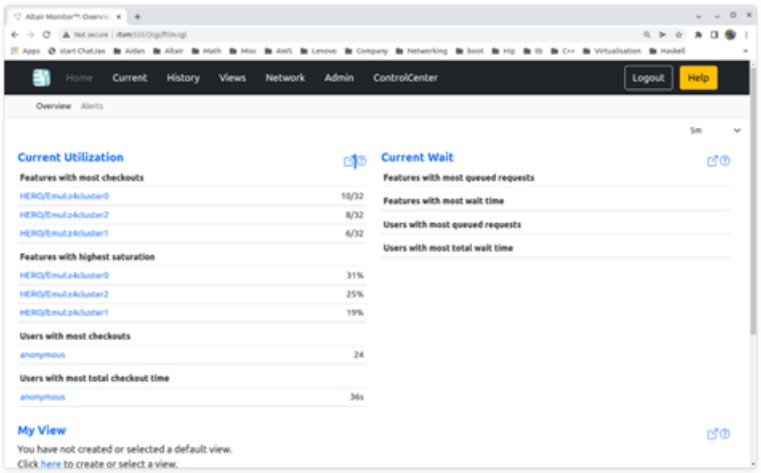

Figure 3:

Remove an Emulator Tasker

If it is necessary to remove an emulator tasker, the following steps should be taken:

1. Stop the tasker.
2. Edit the emulator configuration and remove the tasker from the appropriate RTHOSTS entry.
3. Run the `vovemulmgr config` command.
4. Run the `vovemulmgr config check` command as a check.
   For example, to stop the tasker running on host zbrt15, the following commands could be used:

```
nc cmd vovtaskermgr stop RT_zbrt15
vi <swd>/config/hero/emul.cfg
nc cmd vovemulmgr config
nc cmd vovemulmgr check
```

Hero Commands

hero reserve

Create a reservation on an emulator.

```
hero_reserve: Usage Message

DESCRIPTION:
   Place a reservation on a resource map for HERO.

USAGE:
    % hero_reserve <what> [start] [end] [quantity] [OPTIONS]

OPTIONS:
    -h        -- Show this help.
    -v        -- Increase verbosity. Repeatable.

    -duration -- Duration of the reservation, if an end time is not specified
                 Use "forever" to indicate an expiration that never expires.
    -after    -- Begin time of a future time range pair which gives a span of
                 time in which to attempt to create the reservation
    -before   -- End time of a future time range pair
                 If future time range pair is specified it should be greater
                 than the duration in order to allow a valid window of time in
                 which to place the reservation

    -user     -- Reserve for given list of users
                 (comma separated list)
    -group    -- Reserve for given list of fairshare groups
                 (comma separated list)
    -osgroup  -- Reserve for given list of Unix groups
                 (comma separated list)
    -jobclass -- Reserve for given list of jobclasses
                 (comma separated list)
    -jobproj  -- Reserve for given list of job projects
                 (comma separated list)
    -bucketid -- Reserve for given list of queue buckets
                 (comma separated list)
    -id       -- Reserve for given list of jobs
                 (comma separated list of job ids)

EXAMPLES:
% hero_reserve "HERO:LEAF_sanjoseZ2_(0..11)" -duration 6m -jobproj myProj
% hero_reserve "HERO:LEAF_Z4_cluster0_U(0..7).M(0..7)" -duration 6m -jobproj myProj
% hero_reserve HERO:LEAF_Z4_cluster0_U1.M1 1484612764  forever 1 -jobproj myProj -
group g1,g2
% hero_reserve HERO:Z4_SPAN5 -duration 2h -user John,Bob -after NOW -before 24h
% hero_reserve HERO:Z4_SPAN5 -duration 2h -user John,Bob -after 24h -before 2w
```

vovemulmgr

```
vovemulmgr: Usage Message

USAGE:
    vovemulmgr <SUBCOMMAND> [options]

    Subcommand is case-insensitive.

SUBCOMMAND is one of:
    CHECK                -- Run some configuration checks.
    CONFIG               -- Configure emulators. If there is an error in the
                            configuration then no action will be performed.
    LIST                 -- List the names of emulators.
    LIST_CONFIG_FILES    -- List the emulator configuration files.
    GET_EMUL_PROP        -- Internal debug command.
    GROUP                -- List the names of emulator groups.
    GROUP <group>        -- List the names of emulator in <group>.
    LEAF <emul>          -- List the leaf resources for emulator <emul>.
    ELEMENTS <emul>      -- List the leaf elements for emulator <emul>.
    RTHOSTS <emul>       -- List the runtime hosts for emulator <emul>.
    TYPE <emul>          -- Type of emulator <emul>.
    IGNORE_LIST <emul>   -- List of ignored elements for emulator <emul>.
    TASKER_ENV <emul>    -- Tasker environment for emulator <emul>.
    ADD_PLACEMENT <emul> <target> <reloc-file> [-nowait]
                         -- Add Palladium target for emulator <emul> based
                            on the Cadence relocation file <reloc-file>.
                            The -nowait option will not wait for resource
                            recomputation.
    REMOVE_PLACEMENT <emul> <target> [-nowait]
                         -- Remove a Palladium target for emulator <emul>.
                            The -nowait option will not wait for resource
                            recomputation.
                         --
    LIST_EXTRA_PLACEMENTS
                         -- List add Palladium targets added by emulator.

EXAMPLES:
    % vovemulmgr config
```

Legal Notices

Intellectual Property Rights Notice

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

Altair HyperWorks®, a Design & Simulation Platform

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 HPCWorks®, a HPC & Cloud Platform

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