3DEQUALIZER 4

MHP/Admin

Tutorial, Examples and Reference

Science-D-Visions, 2022-09-05

Contents

1 *Introduction*
- 1.1 Versions and document status
- 1.2 Legal
- 1.3 The purpose of MHP/Admin
- 1.4 Installation
  - 1.4.1 *Running on Linux*
  - 1.4.2 *Running on macOS*
  - 1.4.3 *Running on Windows (status: experimental)*
- 1.5 Notation in this document

2 *Tutorial*
- 2.1 Preliminaries
  - 2.1.1 3DE4/MHP
  - 2.1.2 *Python*
  - 2.1.3 *Secure shell*
- 2.2 Building a network
  - 2.2.1 *The configuration file config.xml*
    - 2.2.1.1 Section `<local>`
    - 2.2.1.2 Section `<hostbase>`
    - 2.2.1.3 Section `<rsh>` (remote shell)
    - 2.2.1.4 Section `<launcher>`
  - 2.2.2 *Launching nodes*
    - 2.2.2.1 Examine the network and clean up (iron fist style)
    - 2.2.2.2 Launching the chief
    - 2.2.2.3 Launching the loyals
- 2.3 Inspecting the network: `describe`, `select`
  - 2.3.1 *Table* `hosts`
  - 2.3.2 *Table* `properties`
- 2.4 Removing and adding nodes
  - 2.4.1 *By* `collect`
  - 2.4.2 *By* `unlink`, `relink`

3 *Examples*
- 3.1 Typical use cases

4 *Reference*
- 4.1 Remote-Shell commands
  - 4.1.1 `launch`-*Syntax*
  - 4.1.2 `run`-*Syntax*
- 4.2 `at`-Syntax
- 4.3 `select...from`-Syntax
  - 4.3.1 `where`-*Clause*
  - 4.3.2 `order`-*Clause*
- 4.4 `describe`-Syntax
- 4.5 Reorganizing the network
  - 4.5.1 `unlink`-*Syntax*
  - 4.5.2 `relink`-*Syntax*

4.5.3 **restart**-*Syntax*
4.5.4 **terminate**-*Syntax*
4.5.5 **make_chief**-*Syntax*
4.5.6 **collect**-*Syntax*
4.6 release-Syntax
4.7 set...to-Syntax
4.8 rescan-Syntax
4.9 UI-Lock related queries
4.9.1 **lock**-*Syntax*
4.9.2 **unlock**-*Syntax*
4.9.3 **relock**-*Syntax*
4.10 Tags and views
4.10.1 **create-tag**-*Syntax*
4.10.2 **drop-tag**-*Syntax*
4.10.3 **create-view**-*Syntax*
4.10.4 **drop-view**-*Syntax*
4.11 show-Syntax
4.12 Shortcuts
A *Style unicode*
Notes

1 *Introduction*

1.1 Versions and document status

Executable version	Compiled	Notes / Changlog
1.0.1	2022-09-05	Release
1.0	2022-07-14	Final, more or less, see notes below
0.7	2022-07-01	Unreleased, bugfixes, style.
0.6	2021-11-08	Unreleased, runs with tde4mhpd 2.0.4
0.0	2021-09-28	Started

The document is currently in flow. The reference part should be complete, but there should more examples. If you are missing something, please let as know (uwe at sci-d-vis.com).

1.2 Legal

We mention a couple of trademarks in this document:

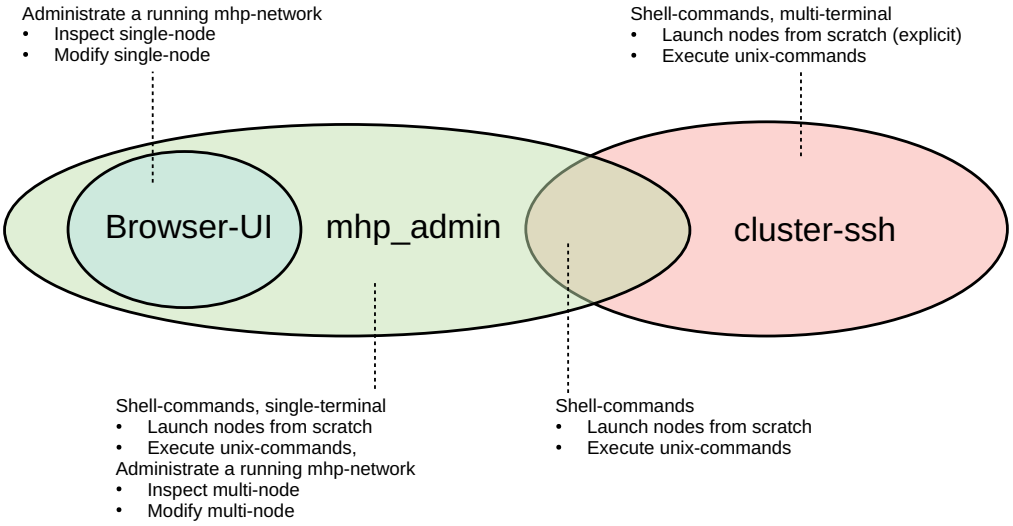
- Linux® is a trademark of The Linux Foundation.
- Windows® and Microsoft® are registered trademarks of the Microsoft Corporation in the U.S. and/or other countries.
- macOS® is a trademark of Apple Inc., registered in the U.S. and other countries and regions.
- "Python" is a trademark of the Python Software Foundation.
- Docker is a trademark or registered trademark of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.
- UNIX® is a trademark of The Open Group.
- POSIX™ is a trademark of The Institute of Electrical and Electronic Engineers, Inc..

The Microsoft Trademarks document does not seem to list WSL2 as a brand or trademark but since it is called Windows Subsystem for Linux, it should be mentioned that it has been developed by Microsoft. Honour to whom honour is due. The owners of the trademarks mentioned in this document are kindly requested to inform us if this list is insufficient or inappropriate.

1.3 The purpose of MHP/Admin

MHP/Admin covers the functions provided by the browser interface and basic functions you may know from cluster administration tools like cluster-ssh, however without output in multiple terminals. The advantage over the browser interface of 3DE4/MHP is, that information about the network can be retrieved for multiple hosts at a time, which makes it easy to detect irregularities in the setup. Also, MHP/Admin can be use re-

motely via a simple terminal without a browser, which is relevant because access from outside is typically restricted. The (slight) advantage over cluster tools is that MHP/Admin provides predefined procedures for launching nodes, while in cluster tools you will typically have to launch the nodes by hand. The disadvantage with respect to cluster tools is, that MHP/Admin is handy only if you set up key-based authentication, whereas (as far as we know) cluster-ssh allows to send passwords to multiple hosts. The following diagram should describe the relationship between the tools:



1.4 Installation

A complete installation of 3DE4 should contain an executable of `mhp_admin` including all dependencies. If you want to move MHP/Admin to a different directory, please make sure that the following files are in the directory:

```
mhp_admin_install_dir
├── clql2.py
├── mhp_admin.py
├── mhp_connector.py
├── sdv_tty_csi.py
├── sdv_tty_table.py
└── sdv_xml_writer.py
```

The executable `mhp_admin` is located in `${3DE4_install dir}/bin`, all other files are located in `${3DE4_install dir}/sysdata/py_scripts`.

1.4.1 *Running on Linux*

`mhp_admin` has been developed under Linux, and we do not expect problems, as long as there is an appropriate version of *Python*. The module `requests` which seems to be a little bit critical on other platforms, is installed out of the box on the Linux distributions we are testing. In case the module is not present, watch out for an `.rpm`- or `.deb`-package named `python3-requests`.

1.4.2 *Running on macOS*

This requires a module named `requests`, which you can install e.g. with `pip3`[(1)]. The installer might complain about security problems if you run it with `sudo`, so invoking it with different options may help here.

```
(sudo) pip3 install --upgrade pip
(sudo) pip3 install requests
```

1.4.3 *Running on Windows (status: experimental)*

We have tested `mhp_admin` under Windows with cygwin. After installing module `requests` like

```
pip3 install --upgrade pip
```

and then suddenly, because `pip` is renamed during update:

```
pip3.9 install --upgrade pip
```

and then surprisingly, because obviously `pip` is moving from one place to another:

```
/usr/local/bin/pip3 install requests
```

We have done only a few tests up to now, but at first sight it seems to work. Bug reports are welcome.

1.5 Notation in this document

The tutorial and the reference manual contain a lot of examples. In many cases I simply run a query on a single machine with ip-address `192.168.178.21` or on the company network. Whenever you find some entry or parameter like `uwe`, just replace it by either your own UNIX-acccount user name, or the name of the user running the MHP-network (i.e. owner of the nodes).

2 *Tutorial*

The tutorial covers setting up the configuration file, launching nodes from scratch and reorganizing the network. Other tasks are explained in the reference part of this document.

2.1 Preliminaries

2.1.1 3DE4/MHP

The minimum version of the 3DE4/MHP daemon tde4mhpd is 2.0.0. Prior versions may be administrated as well, but some functions of `mhp_admin` will not work, and we recommend not to do this.

2.1.2 *Python*

`mhp_admin` works with python version 3.5 or higher. We are testing regularly on python 3.5, 3.7 and 3.8. The tool definitely fails to work with python 3.4 or lower.

2.1.3 *Secure shell*

In this tutorial we will execute commands on remote hosts by means of `ssh`, so please make sure that an `ssh`-daemon is running on each host you would like to participate in the mhp-network. Also, we recommend key-based authentication, in order to avoid entering passwords over and over again. The method is explained on various websites, simply search for "ssh key-based authentication". If you are not in the mood for setting this up right now, you can put it off until later. The machine you are running the administration tool on will of course need the client-side installation of `ssh`.

2.2 Building a network

2.2.1 *The configuration file `config.xml`*

When `mhp_admin` is launched for the first time, a configuration directory `.mhp_admin` is created in the caller's home directory. This directory contains the following files:

```
.mhp_admin
├── config.xml
├── current.xml
├── eula.txt
├── tags.xml
├── uilock.xml
└── views.xml
```

- The file `current.xml` describes the network currently addressed by the tool. Usually there is no need to edit this by hand. `mhp_admin` will modify this upon a query or clause **`at`**, which we will explain in the tutorial.
- The file `tags.xml` contains the specification for color highlighting in `mhp_admin`'s output. Usually you do not edit this by hand, but use the queries explained in section 4.10.1. There is a predefined set of tags, so in principle there is no need to care about this now.
- The file `uilock.xml` is helpful for fast and easy locking the network. You typically edit this file by hand according to your needs. This is explained in section 4.9.1.
- The file `views.xml` contains shortcuts for lengthy queries. You can define these by yourself by means of the query explained in section 4.10.3.

In order to get `mhp_admin` running we must take closer look to file `config.xml`. By this file you specify the basic behaviour of the tool on the client side, the set of hosts you are dealing with, the method(s) for accesing these hosts and finally the methods to launch the 3DE4/MHP-nodes. The structure of the config file is

```
<config>
        <local>...</local>
        <hostbase>...</hostbase>
        <rsh>...</rsh>
        <launcher>...</launcher>
</config>
```

2.2.1.1 Section `<local>`

This section contains an element **`<theme>`** and an optional element **`<style>`** Please set the value of element **`<theme>`** to `dark` or `light`, depend-

ing on your preferred background when working with a terminal. The possible values for element **<style>** are `ascii` and `unicode`. This section in your config file should look like e.g.

```
<config>
        <local>
                <theme>dark</theme>
                <style>unicode</style>
        </local>
<!--- other sections here... -->
</config>
```

All examples in this document are rendered for theme `light` in order to keep it printer-friendly, and style `ascii` since browsers tend to have problems with fancy box diagram glyphs. Terminals which understand ANSI escape sequences for rendering colors or bold/underlined/italic text are widely spread. In the current version, `mhp_admin` makes deep use of these escape sequences. If this turns out to be a problem, please write us (`uwe@sci-d-vis.com`), and we will make color and bold rendering configurable.

2.2.1.2 Section <hostbase>

This section allows to set up sets of ip-addresses, so that commands can be sent to and executed on multiple hosts. The idea behind the host base section is similar to what you might know from tools like cluster-ssh; their configuration file contains a section like

```
clusters = <tag1> <tag2> <tag3>
<tag1> = host1 host2 host3
<tag2> = user@host4 user@host5 host6
<tag3> = <tag1> <tag2>
```

The common idea is to provide a base of hosts to connect to. We recommend to enter all hosts here which might potentially participate in the MHP-network. All hosts should have `ssh`-access and open ports for 3DE4/MHP, typically `57431/tcp` and `57431/udp`. The host sets entered here should not be larger than required, because commands like `run` and `launch` will try to contact **each host** listed here. A **<hostbase>** entry consists of a **<label>** in order to address it and one or more elements **<pattern>** or **<ref>**. A pattern is a single address or a range of addresses, like so:

- **<pattern>**`10.1.1.2`**<pattern>** - Address `10.1.1.2`
- **<pattern>**`10.1.1.3-5`**<pattern>** - Addresses `10.1.1.3`, `...4` and `...5`.
- **<pattern>**`10.1.4-6.99`**<pattern>** - Addresses `10.1.4.99`, `10.1.5.99` and `10.1.6.99`.
- **<pattern>**`10.1.4-6.3-5`**<pattern>** - All addresses like `10.1.i.j` where `i` is 4, 5 or 6 and `j` is 3, 4 or 5.

By means of element **<ref>** you can include one host base into another. Please consider the following example:

```
        <hostbase>
                <label>my_hosts_a</label>
                <pattern>10.1.1.1</pattern>
                <pattern>10.1.1.2</pattern>
                <pattern>10.1.1.3</pattern>
        </hostbase>
        <hostbase>
                <label>my_hosts_b</label>
                <pattern>10.1.1.3-5</pattern>
        </hostbase>
        <hostbase>
                <label>my_hosts_c</label>
                <ref>my_hosts_a</ref>
                <ref>my_hosts_b</ref>
        </hostbase>
```

Host base *`my_hosts_c`* then consists of hosts `10.1.1.1` to `10.1.1.5`. The entire host base for a given label is the union of all addresses matching at least one of the patterns or the host bases referenced. The host bases are rendered as a table by means of the query

```
mhp_admin show hostbases
```

which in our example leads to

```
+-------------------------------------------------------+
|                        host base                      |
+------------+----------------------------------------+
|      label |                              addresses |
+------------+----------------------------------------+
| my_hosts_a |             10.1.1.1, 10.1.1.2, 10.1.1.3 |
| my_hosts_b |                                10.1.1.3-5 |
| my_hosts_c | 10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.3-5 |
+------------+----------------------------------------+
```

2.2.1.3 Section <rsh> (remote shell)

In **<rsh>** you specify how to access remote hosts. Nowadays, `ssh` has become the method of choice, but maybe you decide for something different. A typical section for remote access looks like (in my case):

```
        <rsh>
                <label>my_standard_ssh</label>
                <user>uwe</user>
                <command>ssh :USER:@:HOST: :COMMAND:</command>
        </rsh>
```

which reflects the syntax of the command `ssh` for executing a command on a remote host. The config file contains a reasonable default setup for this case. There is an element **<user>** which represents the default user for remote shell actions. Please enter your preferred user name for remote shell actions there. If you have entered a host base as shown in the previous section you can test immediately if it works:

```
mhp_admin run "id" via my_standard_ssh at 192.168.178.21
```

Equivalently, we can pass a reference to our host base

```
mhp_admin run "id" via my_standard_ssh at my_host_at_home
```

If there is only a single **<rsh>**-entry you can omit the **via**-clause for brevity. If there is more than one entry, the tool will pick the first one. The full syntax is explained in section 4.1.2.

```
mhp_admin run "id" at 192.168.178.21
```

Here, the well-known unix command `id` is executed over ssh at host `192.168.178.21` as user `uwe`. The result in my case is

```
uid=1000(uwe) gid=100(users) groups=100(users),4(adm),24(cdrom),27(sudo),30(dip)...
+-----------------------------+
|          run 'id'           |
+----------------+-------+----+
|           host |  port | rc |
+----------------+-------+----+
| 192.168.178.21 | 57431 |  0 |
+----------------+-------+----+
```

The output consists of the result of `id` and a table showing the return codes for each host the command has been sent to (in our case only a single one). `0` indicates the occurrence of no errors by UNIX-convention. The placeholders `:USER:`, `:HOST:` and `:COMMAND:` in the config file are replaced by the arguments passed in the query, here: `uwe`, `192.168.178.21` and `id`. If this works in your environment you are already done with the **<rsh>**-section.

2.2.1.4 Section <launcher>

If you already have some experience administrating 3DE4/MHP, section **<launcher>** will be easy for you to understand. In your environment you typically launch tde4mhpd from a certain location in the filesystem (most likely the installation directory of 3DE4) with a bunch of options. Consider the following example:

```
        <launcher>
                <label>my_launcher_at_home</label>
                <user>uwe</user>
                <rsh_command>my_standard_ssh</rsh_command>
                <command>/path/to/my/tde4mhpd</command>
        </launcher>
```

Given this entry I can do this:

```
mhp_admin launch my_launcher_at_home at 192.168.178.21
```

This should launch tde4mhpd as user `uwe` at host `192.168.178.21`. The result is the typical output of tde4mhpd

```
+-------------------------------------------------------------------------------------------------------+
|                                              tde4mhp config                                           |
+-------------------+-------------------------------------------------------------------+--------------+
| name              | value                                                             | def'd by     |
+-------------------+-------------------------------------------------------------------+--------------+
| version           | 2.0.3                                                             |              |
| process_mode      | daemon                                                            | cmdline      |
| launched_by       |                                                                   | default      |
| owned_by          | uwe                                                               | default      |
...
```

plus a table which indicates the UNIX return code of the command execution for each host passed at the command line, e.g.

```
+-----------------------------------------------------------------------------------------+
|                                        nodes launched                                   |
+----------------+----+-----+-------------------------------------------------------------+
|           host | rc | msg |                                                     command |
+----------------+----+-----+-------------------------------------------------------------+
| 192.168.178.21 |  0 |  ok | /mnt/server/devel/sdv/privat/uwe/sandbox/uwe/tde4mhpd/tde4mhpd |
+----------------+----+-----+-------------------------------------------------------------+
```

A typical error case occurs when the daemon is already running on the given host:

```
+-------------------------------------------------------------------------------------------------------------+
```

```
|                                                 nodes launched                                                          |
+----------------+----+-------------------------+-------------------------------------------------------------------+
|           host | rc |                     msg |                                                           command |
+----------------+----+-------------------------+-------------------------------------------------------------------+
| 192.168.178.21 |  7 | process already running | /mnt/server/devel/sdv/privat/uwe/sandbox/uwe/tde4mhpd/tde4mhpd |
+----------------+----+-------------------------+-------------------------------------------------------------------+
```

For the full syntax of this query see section 4.1.1.

2.2.2 *Launching nodes*

For this tutorial we will assume that it's our first day in the office after vacation, and the entire network is in a disastrous state. The task is to get it running again. Presumably the most common case in practice is to have a dedicated chief host and a good bunch of "loyals" connecting to this chief host, so this is what we will assume in the following. Our host base is:

```
mhp_admin show hostbases
```

```
+-------------------------------------------------------------------------------------------------+
|                                                host base                                        |
+----------------+--------------------------------------------------------------------------------+
|          label | addresses                                                                      |
+----------------+--------------------------------------------------------------------------------+
|       my_hosts |  10.1.1.24 10.1.1.42 10.1.1.44 10.1.1.46 10.1.1.47 10.1.1.50 10.1.1.103 10.1.1.116 |
| my_hosts_linux |  10.1.1.24 10.1.1.42 10.1.1.44 10.1.1.46 10.1.1.47 10.1.1.50                      |
|   my_hosts_osx |  10.1.1.103 10.1.1.116                                                           |
+----------------+--------------------------------------------------------------------------------+
```

We are going to set up an inhomogenious network from Linux and macOS machines. Our remote shell command is

```
mhp_admin show remote
```

```
+-----------------------------------------------------------+
|                   remote shell commands                   |
+-----------------+------+----------------------------------+
|           label | user |                          command |
+-----------------+------+----------------------------------+
| my_standard_ssh |  uwe | ssh -p 22 :USER:@:HOST: :COMMAND: |
+-----------------+------+----------------------------------+
```

2.2.2.1 Examine the network and clean up (iron fist style)

Let us have a look. First, we should check if all hosts can be reached over secure shell. This is done by sending a simple command which exists on every POSIX-system.

```
mhp_admin run "id" via my_ssh at my_hosts
```

```
uid=753(uwe) gid=100(_lpoperator) groups=100(_lpoperator),12(everyone)...
uid=753(uwe) gid=100(_lpoperator) groups=100(_lpoperator),12(everyone)...
uid=753(uwe) gid=100(users) groups=100(users),27(sudo),121(nopasswdlogin),1003(uwe)
uid=753(uwe) gid=100(users) groups=100(users),4(adm),27(sudo),121(nopasswdlogin)
uid=753(uwe) gid=100(users) groups=100(users),4(adm),27(sudo),753(uwe)
uid=753(uwe) gid=100(users) groups=100(users),10(wheel) ...
uid=753(uwe) gid=1002(uwe) groups=1002(uwe),10(wheel),100(users) ...
uid=753(uwe) gid=1002(uwe) groups=1002(uwe),10(wheel),100(users) ...
+-----------------+
|    run 'id'     |
+------------+----+
|       host | rc |
+------------+----+
| 10.1.1.103 |  0 |
| 10.1.1.116 |  0 |
|  10.1.1.24 |  0 |
|  10.1.1.42 |  0 |
|  10.1.1.44 |  0 |
|  10.1.1.46 |  0 |
|  10.1.1.47 |  0 |
|  10.1.1.50 |  0 |
+------------+----+
```

Ok, now we kill all processes for a fresh start. In our network the executables are called `tde4mhpd` for Linux and `tde4mhpd.osx` for macOS. This may be different in your installation. First, we terminate all processes (please make sure that no one is doing computations on the network):

```
mhp_admin run "killall -9 tde4mhpd" via my_ssh at my_hosts_linux
mhp_admin run "killall -9 tde4mhpd.osx" via my_ssh at my_hosts_osx
```

2.2.2.2 Launching the chief

Now we launch the nodes again and form a network. Our launchers are defined in the launcher-section of the config file. We can write them to the console by

```
mhp_admin show launchers
```

```
+-------------------------------------------------------------------------------+
|                                   launchers                                   |
+-------------------+--------+------+----------------------+----------------------+
|             label | remote | user |              command |              options |
+-------------------+--------+------+----------------------+----------------------+
|   my_launcher_osx | my_ssh |  uwe | /path/to/tde4mhpd.osx | -chief_host 10.1.1.24 |
| my_launcher_linux | my_ssh |  uwe |     /path/to/tde4mhpd | -chief_host 10.1.1.24 |
+-------------------+--------+------+----------------------+----------------------+
```

Your launchers will look different. Most likely, your executables are located in 3DE4's installation directory. The safest way is to launch the chief host `10.1.1.24` first, and then all other nodes:

```
mhp_admin launch my_launcher_linux at 10.1.1.24
```

```
+-----------------------------------------------------------------------------------------+
|                                     nodes launched                                      |
+-----------+-------+----+-----+-----------------------------------------------------------+
|      host |  port | rc | msg |                                                   command |
+-----------+-------+----+-----+-----------------------------------------------------------+
| 10.1.1.24 | 57431 |  0 |  ok | ssh -p 22 uwe@10.1.1.24 /path/to/tde4mhpd -chief_host 10.1.1.24 |
+-----------+-------+----+-----+-----------------------------------------------------------+
```

At this point we have a trivial network consisting of a single node. We should tell `mhp_admin` that this is the network we are working with in the following. This is done by the following query:

```
mhp_admin at 10.1.1.24
```

If you are not sure which network (if any) `mhp_admin` is attached to, just run

```
mhp_admin at
```

and the tool will print something like

```
+-------------------------+
|    current network      |
+-------------+-----------+
| chief_host | 10.1.1.24 |
| chief_port |     57431 |
+-------------+-----------+
```

2.2.2.3 Launching the loyals

Next, we launch the nodes on the Linux- and macOS-boxes, starting for instance with Linux. Our launchers contain options which check in the launched nodes at the chief `10.1.1.24`. (output truncated by hand for better readability):

```
mhp_admin launch my_launcher_linux at my_hosts_linux
```

```
+-------------------------------------------------------------------------------+
|                                   nodes launched                              |
+-----------+----+-------------------------+-------------------------------------+
|      host | rc |                     msg |                             command |
+-----------+----+-------------------------+-------------------------------------+
| 10.1.1.24 |  7 | process already running | /path/to/tde4mhpd -chief_host 10.1.1.24 |
| 10.1.1.42 |  0 |                      ok | /path/to/tde4mhpd -chief_host 10.1.1.24 |
| 10.1.1.44 |  0 |                      ok | /path/to/tde4mhpd -chief_host 10.1.1.24 |
| 10.1.1.46 |  0 |                      ok | /path/to/tde4mhpd -chief_host 10.1.1.24 |
| 10.1.1.47 |  0 |                      ok | /path/to/tde4mhpd -chief_host 10.1.1.24 |
| 10.1.1.50 |  0 |                      ok | /path/to/tde4mhpd -chief_host 10.1.1.24 |
+-----------+----+-------------------------+-------------------------------------+
```

For the chief we get a message "process already running" which is what we can expect, since we have already started the chief. If you want to avoid this message you can set up your host base in a different way and leave out the chief host. Now for the macOS-machines:

```
mhp_admin launch my_launcher_osx at my_hosts_osx
```

```
+-----------------------------------------------------------------+
|                           nodes launched                        |
+------------+----+-----+-------------------------------------------+
|       host | rc | msg |                                   command |
+------------+----+-----+-------------------------------------------+
| 10.1.1.103 |  0 |  ok | /path/to/tde4mhpd.osx -chief_host 10.1.1.24 |
| 10.1.1.116 |  0 |  ok | /path/to/tde4mhpd.osx -chief_host 10.1.1.24 |
+------------+----+-----+-------------------------------------------+
```

We can now check within the MHP-network, if all nodes are running and connected as they should be. The syntax for this query is explained in one of the next sections.

```
mhp_admin select all from hosts at 10.1.1.24
```

The result is:

```
arg (['all'], ['at', '10.1.1.24'])
+-------------------------------------------------------------------------------------------------------------+
|                                     nodes governed by chief 10.1.1.24:57431                                 |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+-----------+---------+------+
|  lock |       host |  port |  hostname |   proc | msg | launched_as |  role | threads |    uptime | version | perf |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+-----------+---------+------+
| False |  10.1.1.24 | 57431 |      guul | daemon |  ok |  standalone | chief |   8 / 0 |  774093.0 |   2.0.4 |  0.0 |
| False |  10.1.1.42 | 57431 |     demon | daemon |  ok |  controlled | loyal |   4 / 4 |  774085.0 |   2.0.4 |  0.0 |
```

```
| False |  10.1.1.44 | 57431 |  belsazar | daemon |  ok |  controlled | loyal |   6 / 6 |  774091.0 |   2.0.4 |  0.0 |
| False |  10.1.1.46 | 57431 |     box01 | daemon |  ok |  standalone | loyal |   9 / 0 |  774090.0 |   2.0.4 |  0.0 |
| False |  10.1.1.47 | 57431 |     box02 | daemon |  ok |  standalone | loyal |   8 / 0 |   1186.48 |   2.0.4 |  0.0 |
| False |  10.1.1.50 | 57431 |     box03 | daemon |  ok |  standalone | loyal |   8 / 0 |   85528.4 |   2.0.4 |  0.0 |
| False | 10.1.1.103 | 57431 |  minimuff | daemon |  ok |  standalone | loyal |   8 / 0 |   87525.1 |   2.0.4 |  0.0 |
| False | 10.1.1.116 | 57431 | applejack | daemon |  ok |  controlled | loyal |   8 / 0 | 1026130.0 |   2.0.4 |  0.0 |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+-----------+---------+------+
|                                 number of nodes listed here - good: 8, bad: 0                                     |
+-----------------------------------------------------------------------------------------------------------------+
```

We now have a running MHP-network, and 3DE4-clients can connect and run computations. All nodes are the same version as it should be.

2.3 Inspecting the network: `describe`, `select`

In principle, the methods for inspecting the network are similar to those you might know from the browser interface: We can list the entire set of hosts, inspect a single host in detail, display the cpu-load when computations are running, check the list of available lens distortion models, render the logfiles in case of problems. The command line tool however allows more control about the representation and also allows to examine properties across hosts, which are not covered by the standard host list representation.

Instead of implementing tons of dash- and double-dash style options we decided to provide a simple query language which reminds a little bit of SQL. We consider each call to `mhp_admin` as a *query*, and the output is usually a mixture of informal messages and a table (as if we were sending a query to an SQL-database from the command line).

Any property of the network is obtained by means of the command **`select`**. We consider the MHP-network as interacting objects the properties of which can be represented as tables. There is a fixed set of tables, e.g. the table of hosts participating in the network, the table of properties for each host, the table of lens distortion models available on each host. In the following we demonstrate how these tables are displayed. The most important one is certainly **`hosts`** because it gives us a quick overview about the relationship between the nodes (e.g. status, mutual linking, role). A possible syntax for **`select`** is

`select` *`columns`* **`from`** *`table`*

where *`columns`* is a space-separated list of identifiers referring to table columns. In this section our network has the following form:

```
+-----------------------------------------------------------------------------------------------------------------+
|                                   nodes governed by chief 10.1.1.24:57431                                       |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+---------+---------+------+
|  lock |       host |  port |  hostname |   proc | msg | launched_as |  role | threads |  uptime | version | perf |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+---------+---------+------+
| False |  10.1.1.46 | 57431 |     box01 | daemon |  ok |  standalone | loyal |   8 / 0 | 8460.42 |   2.0.4 |  0.0 |
| False |  10.1.1.47 | 57431 |     box02 | daemon |  ok |  standalone | loyal |   8 / 0 | 8465.46 |   2.0.4 |  0.0 |
| False |  10.1.1.24 | 57431 |      guul | daemon |  ok |  standalone | chief |   8 / 0 | 48.5919 |   2.0.4 |  0.0 |
| False |  10.1.1.44 | 57431 |  belsazar | daemon |  ok |  controlled | loyal |   6 / 6 |  8465.8 |   2.0.4 |  0.0 |
| False |  10.1.1.42 | 57431 |     demon | daemon |  ok |  controlled | loyal |   4 / 4 | 6522.53 |   2.0.4 |  0.0 |
| False |  10.1.1.50 | 57431 |     box03 | daemon |  ok |  standalone | loyal |   8 / 0 | 8464.44 |   2.0.4 |  0.0 |
| False | 10.1.1.116 | 57431 | applejack | daemon |  ok |  controlled | loyal |   8 / 0 | 262.221 |   2.0.4 |  0.0 |
| False | 10.1.1.103 | 57431 |  minimuff | daemon |  ok |  standalone | loyal |   8 / 0 | 6616.86 |   2.0.4 |  0.0 |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+---------+---------+------+
|                                 number of nodes listed here - good: 8, bad: 0                                     |
+-----------------------------------------------------------------------------------------------------------------+
```

Here we have five standalone nodes and three controlled by 3DE4. Six are running on Linux and two on macOS.

which we obtain by

```
mhp_admin select all from hosts
```

i.e. the columns are specified by *all* and the "table" – to put it in database terminology – is *`hosts`*.

2.3.1 *Table `hosts`*

The table *`hosts`* essentially lists properties collected by and stored at the chief host. The table name has historic reasons. A better name would be *`nodes`*. Instead of listing all nodes we can restrict ourselves to selected columns or rows. Examples:

```
mhp_admin select host status version from hosts
```

writes a table consisting of columns *host status version*:

```
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.24:57431    |
+------------------+-------------+--------------+
|             host |      status |      version |
+------------------+-------------+--------------+
|         10.1.1.24 |          ok |        2.0.4 |
|         10.1.1.44 |          ok |        2.0.4 |
|         10.1.1.46 |          ok |        2.0.4 |
|         10.1.1.47 |          ok |        2.0.4 |
|         10.1.1.42 |          ok |        2.0.4 |
|         10.1.1.50 |          ok |        2.0.4 |
|        10.1.1.116 |          ok |        2.0.4 |
|        10.1.1.103 |          ok |        2.0.4 |
+------------------+-------------+--------------+
| number of nodes listed here - good: 8, bad: 0 |
+-----------------------------------------------+
```

which in many cases is all you are interested in. We can also select rows by a **where**-clause. The syntax for this is

select *columns* **from** *table* **where** *condition*

The **where**-clause is a *Python*-expression which allows to formulate a condition for the table entries. If the condition for a given row/column is not fulfilled, the row is ommited in the output. The query

```
mhp_admin select host hostname launched_as uptime from hosts where "launched_as == 'controlled'"
```

returns information about the nodes controlled by an instance of 3DE4 only.

```
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.24:57431    |
+------------+-----------+-------------+---------+
|       host |  hostname | launched_as |  uptime |
+------------+-----------+-------------+---------+
|  10.1.1.42 |     demon |  controlled | 7215.02 |
|  10.1.1.44 |  belsazar |  controlled |  9158.3 |
| 10.1.1.116 | applejack |  controlled |  954.72 |
+------------+-----------+-------------+---------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

In order to find out the possible columns and their *Python* types (`int`, `float`, `bool`, `string`) you may submit the query **describe** which has the simple syntax

describe *table*

The command

```
mhp_admin describe hosts
```

gives you a list of possible columns of table *hosts* with a short explanation:

```
+-----------------------------------------------------------------------------------------+
|                                      describe hosts                                     |
+----------------------+-------+-------------------------------------------------------+
|               column |  type |                                                 comment |
+----------------------+-------+-------------------------------------------------------+
|                 host |   str |                          Host address in ip4-dot notation |
|                 port |   int |                            Port, default for MHP is 57431 |
|             hostname |   str |                                  Alias for 'sys_hostname' |
|         process_mode |   str |                            Foreground, background, daemon |
|                 proc |   str |                                  Alias for 'process_mode' |
|               status |   str |                    Message from requesting node properties |
|                  msg |   str |                                      Alias for 'status' |
|          launched_by |   str |                    Commandline or some program (e.g. 3DE4) |
|          launched_as |   str |                                    controlled, standalone |
...
```

There is a set of special column specifiers, namely *some*, *all* and *raw*. The query

```
mhp_admin select some from hosts
```

prints a pre-defined selection of columns.

```
mhp_admin select all from hosts
```

prints all columns you see in the browser interface, and

```
mhp_admin select raw from hosts
```

prints out all columns available at all, which is usually not what you want, but we mention this for reasons of completeness.

2.3.2 *Table properties*

While in many cases table *hosts* is all you might be interested in, there is yet another table, *properties* which provides more or at least different information than *hosts* including the number of threads used for computations. The query

```
mhp_admin select launched_by host hostname num_cores num_threads_comm num_threads_priv from properties at all
```

for instance allows you to inspect the number of cpu cores and available threads for each host along with the information whether the node is standalone or launched by a 3DE4-client.

```
+-------------------------------------------------------------------------------------------------------+
|                                                properties                                             |
+-------------+------------+--------------------+-----------+------------------+------------------+---------+
| launched_by |       host |           hostname | num_cores | num_threads_comm | num_threads_priv | request |
+-------------+------------+--------------------+-----------+------------------+------------------+---------+
|             |  10.1.1.24 |               guul |         8 |                8 |                0 |      ok |
|        3DE4 |  10.1.1.42 |              demon |         8 |                4 |                4 |      ok |
|        3DE4 |  10.1.1.44 |           belsazar |        12 |                6 |                6 |      ok |
|             |  10.1.1.46 |              box01 |         8 |                8 |                0 |      ok |
```

```
|             |  10.1.1.47 |              box02 |         8 |                8 |                0 |      ok |
|             |  10.1.1.50 |              box03 |         8 |                8 |                0 |      ok |
|        3DE4 | 10.1.1.116 |          applejack |         8 |                8 |                0 |      ok |
|             | 10.1.1.103 |           minimuff |         8 |                8 |                0 |      ok |
+-------------+------------+--------------------+-----------+------------------+------------------+---------+
```

You can combine this query with a **`where`**-clause in order to filter for hosts with certain properties:

```
mhp_admin select launched_by host hostname uname num_cores from properties at all where "launched_by != ''"
```

which gives

```
+-------------------------------------------------------------------------------------------+
|                                          properties                                       |
+-------------+------------+-----------+-------------------------------+-----------+---------+
| launched_by |       host |  hostname |                         uname | num_cores | request |
+-------------+------------+-----------+-------------------------------+-----------+---------+
|        3DE4 |  10.1.1.42 |     demon | Linux 3.16.0-38-generic x86_64 |         8 |      ok |
|        3DE4 | 10.1.1.116 | applejack |            Darwin 20.3.0 arm64 |         8 |      ok |
|        3DE4 |  10.1.1.44 |  belsazar |  Linux 4.4.0-91-generic x86_64 |        12 |      ok |
+-------------+------------+-----------+-------------------------------+-----------+---------+
```

If you specify a single host you get a similar result as you would expect in the browser interface.

```
mhp_admin select all from properties at 10.1.1.42
```

which leads to

```
+-------------------------------------------------------------------------------------------+
|                                          properties                                       |
+------------------+-------------------------------------------------------------+--------------+
|             name | value                                                       | defd_by      |
+------------------+-------------------------------------------------------------+--------------+
|       executable | tde4mhpd                                                    |              |
|          version | 2.0.4                                                       |              |
|           uptime | 245715                                                      |              |
|              pid | 5410                                                        |              |
|     process_mode | daemon                                                      | default      |
|      launched_by | 3DE4                                                        | cmdline      |
|         owned_by | uwe                                                         | default      |
|             host | 10.1.1.42                                                   | system       |
|             port | 57431                                                       | default      |
|       chief_host | 10.1.1.24                                                   | legacy       |
|       chief_port | 57431                                                       | legacy       |
|     chief_status | ok                                                          |              |
|         is_chief | no                                                          |              |
|     network_mode | explicit                                                    | default      |
|          ip_addr | 10.1.1.42                                                   |              |
|     sys_hostname | demon                                                       |              |
|         subnet_a | 10.1.1.0/24                                                 | default      |
|         subnet_b | 0.0.0.0/0                                                   | undefined    |
|         subnet_c | 0.0.0.0/0                                                   | undefined    |
|  fileserver_base |                                                             | undefined    |
|        num_cores | 8                                                           |              |
| num_threads_priv | 4                                                           | legacy       |
| num_threads_comm | 4                                                           | legacy       |
|      install_dir | /server/devel/sdv/privat/uwe/sandbox/uwe/tde4mhpd           | installation |
|       plugin_dir | <install_dir> / .                                           | default      |
|          tmp_dir | /tmp/                                                       | default      |
|         home_dir | /home/uwe/.3dequalizer/                                     | default      |
|        exec_path | /server/devel/sdv/privat/uwe/sandbox/uwe/tde4mhpd/tde4mhpd | installation |
|      config_file |                                                             | default      |
|          logfile | /tmp/tde4mhp.u753.p57431.log                                | default      |
|         loglevel | LOG_NOTICE                                                  | legacy       |
|            uname | Linux 3.16.0-38-generic x86_64                              |              |
|        is_locked | no                                                          | runtime      |
|     admin_host_a | 0.0.0.0                                                     | runtime      |
|     admin_host_b | 0.0.0.0                                                     | runtime      |
+------------------+-------------------------------------------------------------+--------------+
```

2.4 Removing and adding nodes

In contrast to section 2.2 the operations in this section work on sets of nodes which are already running. Due to maintenance works on your hardware the MHP might dissociate. For instance, let us assume you need to shut down the host running the chief. You terminate the node, shut down the machine, repair what needs to be repaired, reboot and finally restart the node. At this point the chief does not control the MHP-network any longer, because the nodes may have reorganized themselves in order to remain operational. The queries in this section allow to merge the dissociated network fragments into single network easily. From a developer's point of view, the aim of all this is to keep the network operational without implementing too many potentially bug-prone automatics.

2.4.1 *By* `collect`

The button COLLECT in the browser interface has gained a certain popularity, since it gathers disconnected nodes scattered over the local area net-

work in a simple way. Let us assume after a couple of maintenance works on the machines, our MHP-network has shrinked to the following small set of nodes:

```
mhp_admin select all from hosts at 10.1.1.24
```

```
+-----------------------------------------------------------------------------------------------------------------------+
|                                          nodes governed by chief 10.1.1.24:57431                                      |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+----------+---------+------+
|  lock |       host |  port |  hostname |   proc | msg | launched_as |  role | threads |   uptime | version | perf |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+----------+---------+------+
| False |  10.1.1.24 | 57431 |      guul | daemon |  ok |  standalone | chief |   8 / 0 | 247103.0 |   2.0.4 |  0.0 |
| False |  10.1.1.50 | 57431 |     box03 | daemon |  ok |  standalone | loyal |   8 / 0 | 247098.0 |   2.0.4 |  0.0 |
| False | 10.1.1.116 | 57431 | applejack | daemon |  ok |  controlled | loyal |   8 / 0 | 499143.0 |   2.0.4 |  0.0 |
+-------+------------+-------+-----------+--------+-----+-------------+-------+---------+----------+---------+------+
|                                      number of nodes listed here - good: 3, bad: 0                                    |
+-----------------------------------------------------------------------------------------------------------------------+
```

If we know there are nodes running on other machines, we can at least try the following:

```
mhp_admin collect at 10.1.1.24
```

The syntax for this query is quite simple:

collect at *host*

where *host* is any host of the network you want to assemble the nodes in. In my case the output for this query is

```
+-------------------------+
|         collect         |
+--------------------+----+
| status of request | ok |
+--------------------+----+
|          collected |  7 |
+--------------------+----+
|     not authorized |  0 |
|            unknown |  0 |
+--------------------+----+
```

so that

```
mhp_admin select all from hosts
```

now again shows the full set of nodes.

```
+-------------------------------------------------------------------------------------------------------------------------+
|                                          nodes governed by chief 10.1.1.24:57431                                        |
+-------+------------+-------+--------------------+--------+-----+-------------+-------+---------+----------+---------+------+
|  lock |       host |  port |           hostname |   proc | msg | launched_as |  role | threads |   uptime | version | perf |
+-------+------------+-------+--------------------+--------+-----+-------------+-------+---------+----------+---------+------+
| False |  10.1.1.24 | 57431 |               guul | daemon |  ok |  standalone | chief |   8 / 0 | 247002.0 |   2.0.4 |  0.0 |
| False |  10.1.1.44 | 57431 |           belsazar | daemon |  ok |  controlled | loyal |   6 / 6 | 247000.0 |   2.0.4 |  0.0 |
| False |  10.1.1.46 | 57431 |              box01 | daemon |  ok |  standalone | loyal |   8 / 0 | 246999.0 |   2.0.4 |  0.0 |
| False |  10.1.1.47 | 57431 |              box02 | daemon |  ok |  standalone | loyal |   8 / 0 | 246998.0 |   2.0.4 |  0.0 |
| False |  10.1.1.42 | 57431 |              demon | daemon |  ok |  controlled | loyal |   4 / 4 | 246998.0 |   2.0.4 |  0.0 |
| False |  10.1.1.50 | 57431 |              box03 | daemon |  ok |  standalone | loyal |   8 / 0 | 246997.0 |   2.0.4 |  0.0 |
| False | 10.1.1.116 | 57431 |          applejack | daemon |  ok |  controlled | loyal |   8 / 0 | 499041.0 |   2.0.4 |  0.0 |
| False | 10.1.1.103 | 57431 |       minimuff.box | daemon |  ok |  standalone | loyal |   8 / 0 |  2842.14 |   2.0.4 |  0.0 |
+-------+------------+-------+--------------------+--------+-----+-------------+-------+---------+----------+---------+------+
|                                      number of nodes listed here - good: 8, bad: 0                                      |
+-------------------------------------------------------------------------------------------------------------------------+
```

2.4.2 *By* unlink, relink

It is also possible to add one or more disconnected nodes to a network or move them from one network to another. There are two commands for doing this, namely **unlink** and **relink**. The syntax is

unlink *hosts*

for disconnecting nodes from their current network and

relink *hosts* **to** *host*

for moving nodes to a different network. As an example, let us unlink a node from the network:

```
mhp_admin unlink 10.1.1.50
```

Node `10.1.1.50` now forms a network consisting of only a single node, and it is also the chief of this network. Now we move a set of nodes to the new network formed by this node:

```
mhp_admin relink 10.1.1.46 10.1.1.47 to 10.1.1.50
```

so that we now have the following networks:

```
mhp_admin select host hostname role from hosts at 10.1.1.50
```

```
arg (['host', 'hostname', 'role'], ['at', '10.1.1.50'])
+-----------------------------------------------+
```

```
|    nodes governed by chief 10.1.1.50:57431    |
+-----------------+----------------+------------+
|            host |       hostname |       role |
+-----------------+----------------+------------+
|       10.1.1.50 |          box03 |      chief |
|       10.1.1.46 |          box01 |      loyal |
|       10.1.1.47 |          box02 |      loyal |
+-----------------+----------------+------------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

and

```
mhp_admin select host hostname role from hosts at 10.1.1.24
```

```
arg (['host', 'hostname', 'role'], ['at', '10.1.1.24'])
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.24:57431    |
+-----------------+----------------+------------+
|            host |       hostname |       role |
+-----------------+----------------+------------+
|       10.1.1.24 |           guul |      chief |
|       10.1.1.42 |          demon |      loyal |
|       10.1.1.44 |       belsazar |      loyal |
|      10.1.1.116 |      applejack |      loyal |
|      10.1.1.103 |       minimuff |      loyal |
+-----------------+----------------+------------+
| number of nodes listed here - good: 5, bad: 0 |
+-----------------------------------------------+
```

The bottom line is: you can use **`unlink`** and **`relink`** in order to rearrange the network according to your needs. This might be interesting if you want to set up a network temporarily dedicated to some particularly tricky computation job. While everyone else is using the original network, you (as the senior operator) point your 3DE4 client to the new temporary network. When the tricky shot is solved, you reunite the network again, either by **`collect`** or in this case by

```
mhp_admin relink 10.1.1.50 to 10.1.1.24
```

The query **`relink`** is sensitive to the role ("chief", "loyal") of the host passed. In the previous call `10.1.1.50` was the chief of the (small) network. By calling **`relink`** we moved this chief and all its loyal nodes to the new chief `10.1.1.24`, thus reuniting the two networks. **`relink`** applied to a loyal node only moves this loyal node to the new network.

3 *Examples*

3.1 Typical use cases

While the browser interface provides a live-update on the CPU load during computations, `mhp_admin` allows to take at least a snapshot of the load.

```
mhp_admin select all from load
```

In this example three clients are connected and running computations:

```
+-------------------------------------------------------------------------------------------+
|                                        load, snapshot                                     |
+------------+-------+------------+--------+-------------------------------+----------------+
|       host |  port |   hostname | status |                          comm |           priv |
+------------+-------+------------+--------+-------------------------------+----------------+
|  10.1.1.24 | 57431 |       guul |     ok |                               |                |
|  10.1.1.42 | 57431 |      demon |     ok |                               |                |
|  10.1.1.44 | 57431 |   belsazar |     ok |                               |                |
|  10.1.1.46 | 57431 |      box01 |     ok |                               |                |
|  10.1.1.47 | 57431 |      box02 |     ok |                               |                |
|  10.1.1.50 | 57431 |      box03 |     ok |                               |                |
|  10.1.1.52 | 57431 | boxoffice1 |     ok |                               |                |
|  10.1.1.87 | 57431 |   minimuff |     ok |                               |                |
|  10.1.1.88 | 57431 |      mibox |     ok |                               |                |
|  10.1.1.91 | 57431 |     heibox |     ok |                               |                |
| 10.1.1.116 | 57431 |  applejack |     ok |                               |                |
| 10.1.1.118 | 57431 |     azrael |     ok |                               |                |
+------------+-------+------------+--------+-------------------------------+----------------+
```

Another interesting request could be to examine all process on all hosts. If nobody is doing computations on the network, there should not be any computational threads running:

```
select some from processes at all
```

The result in my case is:

```
+-------------------------------------------------------------------+
|                processes at network 10.1.1.24:57431               |
+------------+-------+--------+--------+-------------+------+-------+
|       host |  port |    pid |   ppid |    category | pcpu | request |
+------------+-------+--------+--------+-------------+------+-------+
|  10.1.1.24 | 57431 |    629 |  10537 |        news |  0.0 |      ok |
|            |       |    969 |  10537 |   responder |  0.0 |         |
```

```
|            |       |  10537 |      1 |      parent |  0.0 |         |
|            |       |  10540 |  10537 | updater_udp |  0.0 |         |
|            |       |  32459 |  10537 |        news |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.40 | 57431 |  14777 |      1 |      parent |  0.0 |      ok |
|            |       |  15191 |  14777 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.42 | 57431 |   2001 |  15510 |   responder |  0.0 |      ok |
|            |       |   6570 |      1 |   undefined |  0.0 |         |
|            |       |  15510 |      1 |      parent |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.44 | 57431 |   8517 |      1 |      parent |  0.0 |      ok |
|            |       |  29377 |   8517 | updater_udp |  0.0 |         |
|            |       |  29984 |   8517 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.46 | 57431 |   2745 |  24109 |   responder |  0.0 |      ok |
|            |       |  24109 |      1 |      parent |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.47 | 57431 |  22287 |      1 |      parent |  0.0 |      ok |
|            |       |  29122 |  22287 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.50 | 57431 |   1920 |      1 |      parent |  0.0 |      ok |
|            |       |  14919 |   1920 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.52 | 57431 |    619 |      1 |      parent |  0.0 |      ok |
|            |       |  18137 |    619 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.87 | 57431 |   4310 |      1 |      parent |  0.1 |      ok |
|            |       |   6025 |   4310 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.88 | 57431 | 648033 |      1 |      parent |  0.0 |      ok |
|            |       | 677879 | 648033 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
|  10.1.1.91 | 57431 | 566268 |      1 |      parent |  0.0 |      ok |
|            |       | 575662 | 566268 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
| 10.1.1.116 | 57431 |  67068 |      1 |      parent |  0.0 |      ok |
|            |       |  73155 |  67068 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
| 10.1.1.118 | 57431 |     84 |      1 |      parent |  0.0 |      ok |
|            |       |    578 |     84 |   responder |  0.0 |         |
+------------+-------+--------+--------+-------------+------+---------+
```

We see an undefined process which most likely comes from some undefined operation on the network, but it does not use any CPU time (`pcpu = 0.0`), so there is no need to worry. We see two updater and two news processes which result from a two 3DE4 clients connected to the network. If I run a computation and submit the query

```
mhp_admin select some from processes at all where "category == 'computational'"
```

I get the result

```
+-------------------------------------------------------------------------+
|                   processes at network 10.1.1.24:57431                  |
+------------+-------+--------+--------+---------------+-------+---------+
|       host |  port |    pid |   ppid |      category |  pcpu | request |
+------------+-------+--------+--------+---------------+-------+---------+
|  10.1.1.42 | 57431 |   1975 |  15510 | computational |   105 |      ok |
|  10.1.1.47 | 57431 |  29081 |  22287 | computational |   283 |      ok |
|  10.1.1.52 | 57431 |  18012 |    619 | computational |   389 |      ok |
| 10.1.1.116 | 57431 |  73139 |  67068 | computational | 782.9 |      ok |
|  10.1.1.87 | 57431 |   6009 |   4310 | computational | 785.9 |      ok |
|  10.1.1.24 | 57431 |    866 |  10537 | computational |   861 |      ok |
|  10.1.1.88 | 57431 | 677857 | 648033 | computational |   446 |      ok |
|  10.1.1.91 | 57431 | 575637 | 566268 | computational |   493 |      ok |
|  10.1.1.46 | 57431 |   2703 |  24109 | computational |   676 |      ok |
|  10.1.1.50 | 57431 |  14885 |   1920 | computational |   649 |      ok |
|  10.1.1.44 | 57431 |  29750 |   8517 | computational |  1012 |      ok |
|  10.1.1.40 | 57431 |  15166 |  14777 | computational |   679 |      ok |
| 10.1.1.118 | 57431 |    545 |     84 | computational |  2235 |      ok |
+------------+-------+--------+--------+---------------+-------+---------+
```

In case a process gets out of control and continues to run, we can terminate it with the following command (where 545 is only an example):

```
run "kill 545" at 10.1.1.118
```

Of course, this is a situtation we are trying to avoid in tde4mhpd by any means necessary, but we cannot exclude it completely, therefore it makes sense to have a method for terminating processes by hand.

4 Reference

In this section we document all queries with examples. Most of the queries have different forms which we list at the beginning of each subsection. Placeholders are written like e.g.

- *`hosts`*, *`columns`*, *`command`*
- *`where-clause`*, *`order-clause`*

Keywords of the query language are written as

- **`select`**, **`describe`**, **`set`**
- **`if`**, **`where`**
- **`as`**, **`at`**, **`by`**, **`for`**, **`from`**, **`to`**, **`via`**

- **all**, **range**

Table names are not considered as keywords, and we write them as

- *hosts*, *properties*, *lens_models*
- *hostbases*, *views*

Explicit property or column names are written as

- *host*, *port*, *lock*
- *num_threads_comm*, *num_threads_priv*

4.1 Remote-Shell commands

The commands in this section differ from the others in that they are not based on an existing MHP-network. On the contrary, these queries are used to create an MHP-network from scratch. Thus, they do not rely on lists of hosts known to a running node, but on the host bases defined in the configuration file.

4.1.1 **launch**-*Syntax*

Form

- **launch** *launcher* **at** *hosts* [**options** "*options*"]?
- **launch** *launcher* **at** *hostbase* [**options** "*options*"]?

Description

This query starts the MHP-node via a remote shell command according to the specification in the config file, on the hosts specified by the **at**-clause a single host or on multiple hosts. The ip-addresses may passed explicitly or as a reference to a host base. If the **options**-clause is given, options specified in the config file are ignored, and only *options* from the clause are passed to the executable.

Examples

The following examples are based on a launcher labelled `my_launcher_linux` and a host base labelled `my_hosts_linux`. Launch nodes on the hosts passed explicitly:

```
mhp_admin launch my_launcher_linux at 10.1.1.24 10.1.1.44 10.1.1.50
```

Launch nodes on the hosts defined in the host base:

```
mhp_admin launch my_launcher_linux at my_hosts_linux
```

Launch node and override options in the config file:

```
mhp_admin launch my_launcher_linux at 10.1.1.44 options "-chief_host 10.1.1.50"
```

See also

- 2.2.1 The configuration file

4.1.2 **run**-*Syntax*

Form

- **run** *command* [**via** *rsh_label*]? [**as** *user*]? **at** *hosts*
- **run** *command* [**via** *rsh_label*]? [**as** *user*]? **at** **all**
- **run** *command* [**via** *rsh_label*]? [**as** *user*]? **at** *hostbase*

Description

Execute any UNIX-command with parameters described by the string *command* via the remote shell command given by *rsh_label* as user *user* on the hosts specified. If no **via**-clause is given, the first **<rsh>**-entry in the config file is used for running the remote shell. If no **as**-clause is given, the content of **<user>** in the **<rsh>**-entry is the user running the remote shell. Typically, this query is used in order to examine hosts before launching MHP-nodes, or in order to terminate MHP-nodes the hard way if the network is in some desolate state and needs a fresh start. Handle with care and **think twice** before you run a query of this type particularly if you are running your command as `root`! If you are not sure about the set of hosts affected, run a dummy command like `id` or `hostname` first. and have a look at the output.

Examples

Examples are based on a host base labelled `my_hosts`.

Check if we can connect to all hosts by our remote shell command:

```
mhp_admin run "id" via my_standard_ssh at my_hosts
```

Check on which host a node is running:

```
mhp_admin run "ps -e | grep tde4mhp" via my_standard_ssh at my_hosts
```

Terminate all nodes unconditionally. All computations are aborted.

```
mhp_admin run "killall -9 tde4mhpd" via my_standard_ssh at my_hosts
```

Shortest form to test remote shell access if there is only one single **<rsh>**-entry in the config file (which is usually the case):

```
mhp_admin run "id" at 10.1.1.50
```

A trick for getting a more structured output is to run a specially crafted `echo` ahead of the command, so that we see which output come from which host:

```
mhp_admin run "echo -n ':USER:@:HOST:: ';id" at 10.1.1.50
```

which in a network with three hosts leads to:

```
uwe@10.1.1.46: uid=753(uwe) gid=100(users) groups=100(users),10(wheel) ...
uwe@10.1.1.47: uid=753(uwe) gid=1002(uwe) groups=1002(uwe),10(wheel),100(users) ...
uwe@10.1.1.50: uid=753(uwe) gid=1002(uwe) groups=1002(uwe),10(wheel),100(users) ...
+-------------------------------------+
| run 'echo -n ':USER:@:HOST:: ';id' |
+---------------------+--------------+
|                host |           rc |
+---------------------+--------------+
|           10.1.1.46 |            0 |
|           10.1.1.47 |            0 |
|           10.1.1.50 |            0 |
+---------------------+--------------+
```

See also

- 2.2.1 The configuration file

4.2 `at`-Syntax

Form

- **`at`** *`node`*
- **`at`**

Description

- *`node`*: Select a network by passing one of its nodes. The network then becomes the "current" network. The information about the current network is stored on the client side, i.e. in `mhp_admin`'s file `~/.mhp_admin/current.xml`. You can edit this file by hand or modify it by means of other tools, but please make sure that permissions are `rw-------` for security reasons.
- without paramerers: Show the current network.

Examples

Given a network of three nodes `10.1.1.46`, `10.1.1.47`, `10.1.1.50` with chief `10.1.1.50`, passing any of these addresses will make this the current network:

```
mhp_admin at 10.1.1.47
```

The tool determins the chief (which represents the network) by querying the properties of the node passed.

```
+-----------------------------------------------+
| request http://10.1.1.47:57431/properties.xml |
+------------------------+----------------------+
|      status of request |                   ok |
+------------------------+----------------------+
|        chief host:port |      10.1.1.50:57431 |
+------------------------+----------------------+
```

Sending **`at`** without parameters

```
mhp_admin at
```

shows the current network without roundtrip to the nodes, by reading from the file `~/.mhp_admin/current.xml`:

```
+-----------------------+
|    current network    |
+------------+----------+
| chief_host | 10.1.1.50 |
| chief_port |     57431 |
+------------+----------+
```

4.3 select...from-Syntax

Form

- **select** *columns* **from** { *hosts* | *load* | *users* } [**at** *network*]? *where-clause*? *order-clause*?
- **select** *columns* **from** { *properties* | *logfile* | *processes* | *lens_models* | *skills* } **at** { **all** | *hosts* } *where-clause*? *order-clause*?
- **select** *columns* **from** *hostbase* *hostbase*
- **select** *columns* **from** *lens_model* *lens_model* **at** *host*
- **select** *columns* **from** *skill* *skill* **at** *host*

Description

- { *hosts* | *load* | *users* }: The **at**-clause in this case refers to a single node representing the network of interest. If no **at**-clause is given, the table for the current network is shown. If an **at**-clause is given, the network passed at the **at**-clause becomes the current one (check with `mhp_admin at`). Tables *hosts*, *load* and *users* correspond mode or less to the pages `hosts.html`, `load.html` and `users.html` in the browser interface.
- { *properties* | *logfile* | *processes* | *lens_models* | *skills* }: The **at**-clause selects **all** nodes in the MHP-network or a list of nodes, separated by white space and/or ranges of ip-addresses, like **range** `10.1.1.44 10.1.1.47`. All addresses and address ranges passed here are clipped against the set of nodes of the MHP-network.
- *hostbase*: *hostbase* refers to one of the host bases you have defined in `~/.mhp_admin/config.xml`.
- *lens_model*: *lens_model* refers to one of the lens distortion models available at the node passed.
- *skill*: *skill* refers to one of the skills available at the node passed. This is for future use. Currently there is only one single skill, namely `ph4_core` core computations for 3DE4 (although a plugin-concept is already implemented and ready to use).

columns is either one of { **raw** | **all** | **some** } or a white space separated list of column names. **raw** prints out all columns in the form they are delivered by tde4mhpd, which might be not very comprehensible. **all** refers to a representation which is quite close to the representation in the browser interface. **some** refers to a selection of columns which we found to be interesting and sufficient in many situations, the "developer's choice", if you like. Often, you are interested in a different set of properties, and in this case you simply name the columns you would like to see.

Examples

List some properties of all nodes of the current network.

```
mhp_admin select some from hosts
```

List properties of a single node. The representation for a single host is similar to the one in the browser interface: one column for the property name, one for the value, and one which describes how the value has been defined:

```
mhp_admin select all from properties at 10.1.1.44
```

List ip-address and the OS the nodes ranging from `10.1.1.46` to `10.1.1.50` are running on. *uname* refers to the UNIX-command `uname`. Semantically, this type of query is similar to the *hosts*-query. The latter, however, is only able to devliver a small set of properties, disposed at the chief, while the *properties*-query does a round trip to each node specified in the **at**-clause and provides more detailed information.

```
mhp_admin select host uname from properties at range 10.1.1.46 10.1.1.50
```

Show the current CPU load for the current network, similar to the browser interface but as a snapshot without perodic refreshing.

```
mhp_admin select all from load
```

Recall hosts in one of the host bases defined in `~/.mhp_admin/config.xml`. This is a client side request without roundtrip to the nodes.

```
mhp_admin select all from hostbase my_hosts_linux
```

Show lens distortion plugins available at each node:

```
mhp_admin select some from lens_models at all where "linking == 'plugin'"
```

See also

- 4.4 **`describe`**-Syntax
- 4.3.1 **`where`**-Clause
- 4.3.2 **`order`**-Clause
- 2.2.1 The configuration file

4.3.1 `where`-*Clause*

Form

- **`where`** "*`boolean_python_expression`*"

Description

This clause restricts the output of any query to table rows for which the boolean expression is `true`. Please note that you need quotation marks around the expression, since operators like `<`, `>`, `*`, `!` etc. will be interpreted by the shell and not passed as arguments to the query.

Examples

List all nodes which have a version number other than e.g. `2.0.4`, e.g. in order to detect version mismatch.

```
mhp_admin select host version from hosts where "version != '2.0.4'"
```

List all nodes running on macOS

```
mhp_admin select host uname from properties at all where "'Darwin' in uname"
```

List all nodes controlled by a 3DE4-instance

```
mhp_admin select host from properties at all where "launched_by == '3DE4'"
```

See also

- 4.3 **`select...from`**-Syntax

4.3.2 `order`-*Clause*

Form

- **`order by`** [*`column`* { **`ascending`** | **`descending`** }]+

Description

By this clause, which appears in **`select...from`**-queries, table rows are sorted according to the columns and direction specified. The clause is applied from the right to the left, i.e. order criteria on the left have a higher priority than order criteria on the right.

Examples

The output of

```
mhp_admin select all from hosts order by role ascending launched_as descending
```

will put the "chief" to the first row, because it is lexically lower than "loyal". The loyals will be sorted according to their property *`launched_as`*: `standalone` above, `controlled` below.

See also

- 4.3 **`select...from`**-Syntax

4.4 `describe`-Syntax

Form

- **`describe`** { *`hosts`* | *`lens_model`* | *`lens_models`* | *`load`* | *`logfile`* | *`processes`* | *`properties`* | *`skill`* | *`skills`* | *`users`* | *`hostbase`* }

Description

Print a table, which describes the output of the **`select...from`**-query for the table passed to **`describe`**, i.e. name and data type for each column, and hopefully a comment.

Examples

```
mhp_admin describe lens_model
```

```
+-------------------------------------------------------+
|                   describe lens_model                 |
+---------+------+---------------------------------------+
|  column | type |                               comment |
+---------+------+---------------------------------------+
|    name |  str |                  Name of lens parameter |
|    type |  str |                  Type of lens parameter |
| default |  str | Default value (mostly adjustable double) |
|   range |  str |         Range for very coarse adjustment |
+---------+------+---------------------------------------+
```

See also

- 4.3 **`select...from`**-Syntax

4.5 Reorganizing the network

By means of the queries in this section you can split the network into smaller ones or merge multiple networks into a single one. The query **`collect`** is often used for repairing disintegrated networks.

4.5.1 `unlink`-*Syntax*

Form

- **`unlink`** *`nodes`*
- **`unlink all`**

Description

Unlink loyal nodes from their network. After this query each of the nodes forms a network consisting of only the node itself. If the query is applied to the chief of a network, the tool will respond something like "nothing to do" because the chief cannot be unlinked from its own network.

Examples

```
mhp_admin unlink 10.1.1.46 10.1.1.47
```

Unlink nodes `10.1.1.46` and `10.1.1.47` from the current network.

```
mhp_admin unlink all
```

Unlink all nodes from the current network. This leads to a totally disintegrated set of monadic networks. You definitly want to send the query **`collect`** afterwards. We assume that this form is not really useful in practice, but we have implemented it for reasons of completeness and debugging purposes.

See also

- 4.5.2 **`relink`**-Syntax.
- 4.5.6 **`collect`**-Syntax.
- 4.5.5 **`make_chief`**-Syntax.

4.5.2 `relink`-*Syntax*

Form

- **`relink`** *`nodes`* **`to`** *`node`*

Description

Relink nodes from one network to another. *`node`* is the chief of the target network. This query is useful for dealing with more than one network, or for repairing a disintegrated network by hand. The operation is refused if the caller of `mhp_admin` is not authorized to access any of the affected nodes due to ui-locking. The node being relinked accepts the lock-state of the target network.

Examples

Given a network with chief `10.1.1.24` and loyals `10.1.1.46`, `10.1.1.47`, `10.1.1.50`. We can split this network into two parts by

```
mhp_admin unlink 10.1.1.50
mhp_admin relink 10.1.1.47 to 10.1.1.50
```

which leads to two networks each consisting of two nodes:

- `10.1.1.24` (chief), `10.1.1.46`
- `10.1.1.50` (chief), `10.1.1.47`

The two networks can be re-merged into a single one for instance by

```
mhp_admin relink 10.1.1.50 to 10.1.1.24
```

with chief `10.1.1.24`, or by

```
mhp_admin relink 10.1.1.24 to 10.1.1.50
```

with chief `10.1.1.50`.

See also

- 4.5.1 **`unlink`**-Syntax.

4.5.3 `restart`-*Syntax*

Form

- **`restart`** *`hosts`*
- **`restart all`**

Description

The nodes specified by *`hosts`* in the current network are restarted from the same executable path as they were originally started from. It is not quite clear in how far this query is useful for administration. It definitely has its flaws, but we are using it after compiling to quickly get a new version running, instead of rebuilding the entire network.

4.5.4 `terminate`-*Syntax*

Form

- **`terminate`** *`hosts`*
- **`terminate all`**

Description

The nodes specified by *`hosts`* in the current network are terminated in a well-defined way. This is the recommended method disconnecting a node, e.g. if you need to shut down or reboot the host for maintenance.

Examples

Given a network

```
mhp_admin select host port role from hosts
```

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.50:57431    |
+------------------+--------------+-------------+
|             host |         port |        role |
+------------------+--------------+-------------+
|        10.1.1.50 |        57431 |       chief |
|        10.1.1.47 |        57431 |       loyal |
|        10.1.1.46 |        57431 |       loyal |
+------------------+--------------+-------------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

we terminate a node

```
mhp_admin terminate 10.1.1.50
```

and are left with a well-defined network:

```
mhp_admin select host port role from hosts
```

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+------------------+--------------+-------------+
|             host |         port |        role |
+------------------+--------------+-------------+
|        10.1.1.46 |        57431 |       chief |
|        10.1.1.47 |        57431 |       loyal |
+------------------+--------------+-------------+
| number of nodes listed here - good: 2, bad: 0 |
+-----------------------------------------------+
```

If instead we kill the process and its child processes the hard way, like

```
mhp_admin run "killall -9 tde4mhpd" at 10.1.1.50
```

```
arg: ('killall -9 tde4mhpd', ['10.1.1.50'])
+---------------------------+
| run 'killall -9 tde4mhpd' |
+-----------------+---------+
|            host |      rc |
+-----------------+---------+
|       10.1.1.50 |       0 |
+-----------------+---------+
```

the network gets into an undefined state, which is not what we want:

```
mhp_admin select host port role from hosts
```

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+-----------------+-------------+---------------+
|            host |        port |          role |
+-----------------+-------------+---------------+
|       10.1.1.46 |       57431 |         chief |
|       10.1.1.47 |       57431 |         loyal |
|       10.1.1.50 |       57431 |       unclear |
+-----------------+-------------+---------------+
| number of nodes listed here - good: 2, bad: 1 |
+-----------------------------------------------+
```

and this is why we recommend to send the **`terminate`**-query instead. In this example we have terminated a loyal node. If you need to terminate the chief we recommend to move the chief role to a different node, using the query **`make_chief`**, see section 4.5.5.

See also

- 4.1.2 **`run`**-Syntax
- 4.6 **`release`**-Syntax

4.5.5 `make_chief`-*Syntax*

Form

- **`make_chief`** *`host`*

Description

Assign the chief role to a different node. This should be done if the chief is going to be terminated, e.g. for maintenance. The network remains operational, and core computations are not interrupted.

Examples

In the network

```
mhp_admin select host port role from hosts
```

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.47:57431    |
+------------------+--------------+-------------+
|             host |         port |        role |
+------------------+--------------+-------------+
|        10.1.1.47 |        57431 |       chief |
|        10.1.1.46 |        57431 |       loyal |
|        10.1.1.50 |        57431 |       loyal |
+------------------+--------------+-------------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

we would like to terminate the chief. The smoothest way to do this is to transfer the chief role to a different node, like e.g.

```
mhp_admin make_chief 10.1.1.46
```

and then

```
mhp_admin terminate 10.1.1.47
```

The remaining network is

```
mhp_admin select host port role from hosts
```

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+------------------+--------------+-------------+
|             host |         port |        role |
+------------------+--------------+-------------+
|        10.1.1.46 |        57431 |       chief |
|        10.1.1.50 |        57431 |       loyal |
+------------------+--------------+-------------+
| number of nodes listed here - good: 2, bad: 0 |
+-----------------------------------------------+
```

See also

- 4.5.4 **`terminate`**-Syntax

4.5.6 `collect`-*Syntax*

Form

- **`collect at`** *`host`*

Description

The hosts known to the node at *`host`* are scanned for running instances of `tde4mhpd`. Each instance found is then incorporated into the `MHP`-network. A node will refuse the request if the caller of `mhp_admin` is not authorized to access the node due to ui-locking. If the network specified by *`host`* is locked, each node collected becomes locked as well. Collecting only works for nodes listening to the same port.

Examples

Given a network

```
mhp_admin select host port role from hosts at 10.1.1.24
```

```
arg (['host', 'port', 'role'], ['at', '10.1.1.24'])
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.24:57431    |
+-------------------+-------------+-------------+
|              host |        port |        role |
+-------------------+-------------+-------------+
|         10.1.1.24 |       57431 |       chief |
|         10.1.1.42 |       57431 |       loyal |
|         10.1.1.44 |       57431 |       loyal |
|         10.1.1.46 |       57431 |       loyal |
|         10.1.1.47 |       57431 |       loyal |
|         10.1.1.50 |       57431 |       loyal |
|        10.1.1.103 |       57431 |       loyal |
+-------------------+-------------+-------------+
| number of nodes listed here - good: 7, bad: 0 |
+-----------------------------------------------+
```

we completely disintegrate this for demonstration purposes

```
mhp_admin unlink all
```

which leads to seven monadic networks, and our chief node does not have loyals any longer:

```
mhp_admin select host port role from hosts at 10.1.1.24
```

Now we collect all nodes at `10.1.1.24`

```
mhp_admin collect at 10.1.1.24
```

```
+------------------------+
|        collect         |
+-------------------+----+
| status of request | ok |
+-------------------+----+
```

```
|         collected |  6 |
+-------------------+----+
|    not authorized |  0 |
|           unknown |  0 |
+-------------------+----+
```

and finally get the network we had before splitting it up:

```
mhp_admin select host port role from hosts
```

```
arg (['host', 'port', 'role'], ['at', '10.1.1.24'])
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.24:57431    |
+-------------------+-------------+-------------+
|              host |        port |        role |
+-------------------+-------------+-------------+
|         10.1.1.24 |       57431 |       chief |
|         10.1.1.42 |       57431 |       loyal |
|         10.1.1.44 |       57431 |       loyal |
|         10.1.1.46 |       57431 |       loyal |
|         10.1.1.47 |       57431 |       loyal |
|         10.1.1.50 |       57431 |       loyal |
|        10.1.1.103 |       57431 |       loyal |
+-------------------+-------------+-------------+
| number of nodes listed here - good: 7, bad: 0 |
+-----------------------------------------------+
```

See also

- 4.5.1 **`unlink`**-Syntax.

4.6 `release`-Syntax

Form

- **`release`** *`nodes`*

Description

`nodes` is a node or list of nodes, separated by white space. The purpose of this command is to remove node entries from the node list in table *`hosts`* in case they are in an undefined state, e.g. after a hard termination.

Examples

As an example consider the result of `killall -9 tde4mhpd` in section 4.5.4. The host list contains an entry for `10.1.1.50`, but the node on this host does not exist any longer, since due to the command `killall` it did not properly check out at the chief node.

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+-----------------+-------------+---------------+
|            host |        port |          role |
+-----------------+-------------+---------------+
|       10.1.1.46 |       57431 |         chief |
|       10.1.1.47 |       57431 |         loyal |
|       10.1.1.50 |       57431 |       unclear |
+-----------------+-------------+---------------+
| number of nodes listed here - good: 2, bad: 1 |
+-----------------------------------------------+
```

In this situation you can remove the undefined entry by

```
mhp_admin release 10.1.1.50
```

and the network is now well-defined:

```
arg (['host', 'port', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+------------------+--------------+-------------+
|             host |         port |        role |
+------------------+--------------+-------------+
|        10.1.1.46 |        57431 |       chief |
|        10.1.1.47 |        57431 |       loyal |
+------------------+--------------+-------------+
| number of nodes listed here - good: 2, bad: 0 |
+-----------------------------------------------+
```

Another undefined entry may occur if a node is running, but it is not checked in at the chief, while the chief's node list says it is checked in. These node are labelled `renegade`. This situation can result from inappropriate termination and relaunch of a loyal or the chief, or maybe **`unlink`** or **`relink`** operations in an overloaded network. One way to fix this is again to send query **`release`**, or to **`relink`** the renegade to the

chief.

4.7 `set...to`-Syntax

Form

- **set** *properties* **at** *hosts* **to** *values*
- **set** *properties* **at all to** *values*

Description

Set one or more properties to given values at the hosts specified. The number of values in the **to**-clause must coincide with the number of properties. Currently there are not many properties to set at runtime:

- *num_threads_comm* - number of threads available to public
- *num_threads_priv* - number of threads available to the user working on the host (controlled by 3DE4)
- *loglevel* - 0: print almost nothing to the logfile, 7: print a lot

Examples

A typical setup for a standalone node with 8 cpu-cores would be 8 community threads and 0 private threads, but maybe you want to try out having one or two threads more in order to avoid idle times:

```
mhp_admin set num_threads_comm num_threads_priv at 10.1.1.46 10.1.1.47 to 9 0
```

When you experience trouble in setting up a network (and contact us) it might be helpful to set the logging level higher, like so:

```
mhp_admin set loglevel at all to 7
```

and send us the logfiles by email. In the ideal case you will not need to care about this property. Let us have a look.

```
mhp_admin select num_threads_comm num_threads_priv loglevel from properties at 10.1.1.46
```

For this single node query the result looks like

```
+----------------------------------------+
|               properties               |
+------------------+-----------+---------+
| name             |     value | defd_by |
+------------------+-----------+---------+
| num_threads_priv |         0 |  legacy |
| num_threads_comm |         9 |  legacy |
| loglevel         | LOG_DEBUG |  legacy |
+------------------+-----------+---------+
```

while for the multinode query

```
mhp_admin select host num_threads_comm num_threads_priv loglevel from properties at 10.1.1.46 10.1.1.47
```

we have

```
(['host', 'num_threads_comm', 'num_threads_priv', 'loglevel'], ['10.1.1.46', '10.1.1.47'])
+-----------------------------------------------------------------------+
|                                properties                             |
+-----------+------------------+------------------+-----------+---------+
|      host | num_threads_comm | num_threads_priv |  loglevel | request |
+-----------+------------------+------------------+-----------+---------+
| 10.1.1.46 |                9 |                0 | LOG_DEBUG |      ok |
| 10.1.1.47 |                9 |                0 | LOG_DEBUG |      ok |
+-----------+------------------+------------------+-----------+---------+
```

The output for *loglevel* is always a symbolic constant. We might add a way to pass symbolic constants to the query **set...to** in the future.

4.8 `rescan`-Syntax

Form

- **rescan** *nodes*
- **rescan all**

Description

Plugins for lens distortion (and later skills) can be added as functionality to a node without restarting it. This query causes the nodes to rescan their plugin directories so that plugins become available for computations.

Examples

Consider a set of nodes with the following plugin directories (paths edited for the sake of brevity):

```
mhp_admin select host install_dir plugin_dir from properties at all
```

```
(['host', 'install_dir', 'plugin_dir'], 'all')
+-------------------------------------------------------------+
|                          properties                         |
+-----------+----------------------+-------------------+-------+
|      host |          install_dir |        plugin_dir | request |
+-----------+----------------------+-------------------+-------+
| 10.1.1.46 | /server/.../tde4mhpd | <install_dir> / . |      ok |
| 10.1.1.47 | /server/.../tde4mhpd | <install_dir> / . |      ok |
| 10.1.1.50 | /server/.../tde4mhpd | <install_dir> / . |      ok |
+-----------+----------------------+-------------------+-------+
```

The lens distortion models currently available at the nodes are:

```
mhp_admin select some from lens_models at all
```

```
+-------------------------------------------------------------------------------------------+
|                              lens_models at network 10.1.1.46:57431                       |
+-----------+-------+-------------------------------------------------------+---------+---------+
|      host |  port |                                                  name | linking | request |
+-----------+-------+-------------------------------------------------------+---------+---------+
| 10.1.1.46 | 57431 |                                  3DE Classic LD Model |  static |      ok |
|           |       |                    3DE4 Anamorphic - Rescaled, Degree 4 |  static |         |
|           |       |                    3DE4 Anamorphic - Standard, Degree 4 |  static |         |
|           |       |                              3DE4 Anamorphic, Degree 6 |  static |         |
|           |       |                         3DE4 Radial - Fisheye, Degree 8 |  static |         |
|           |       |                        3DE4 Radial - Standard, Degree 4 |  static |         |
+-----------+-------+-------------------------------------------------------+---------+---------+
| 10.1.1.47 | 57431 |                                  3DE Classic LD Model |  static |      ok |
|           |       |                    3DE4 Anamorphic - Rescaled, Degree 4 |  static |         |
|           |       |                    3DE4 Anamorphic - Standard, Degree 4 |  static |         |
|           |       |                              3DE4 Anamorphic, Degree 6 |  static |         |
|           |       |                         3DE4 Radial - Fisheye, Degree 8 |  static |         |
|           |       |                        3DE4 Radial - Standard, Degree 4 |  static |         |
+-----------+-------+-------------------------------------------------------+---------+---------+
| 10.1.1.50 | 57431 |                                  3DE Classic LD Model |  static |      ok |
|           |       |                    3DE4 Anamorphic - Rescaled, Degree 4 |  static |         |
|           |       |                    3DE4 Anamorphic - Standard, Degree 4 |  static |         |
|           |       |                              3DE4 Anamorphic, Degree 6 |  static |         |
|           |       |                         3DE4 Radial - Fisheye, Degree 8 |  static |         |
|           |       |                        3DE4 Radial - Standard, Degree 4 |  static |         |
+-----------+-------+-------------------------------------------------------+---------+---------+
|                                 number of entries listed here: 21                         |
+-------------------------------------------------------------------------------------------+
```

Now assume we have a lens distortion plugin (e.g. one of the experimental plugins from the LDPK), and we would like to add it to 3DE4/MHP. We copy it to the plugin directory:

```
cp /path/to/tde4_ldp_radial_fisheye_stereographic_deg_8.so /server/.../tde4mhpd/
```

and have the nodes rescan:

```
mhp_admin rescan all
```

Then we see that plugins are now available:

```
mhp_admin select some from lens_models at all where "linking == 'plugin'"
```

```
+-------------------------------------------------------------------------------------------+
|                              lens_models at network 10.1.1.46:57431                       |
+-----------+-------+-------------------------------------------------------+---------+---------+
|      host |  port |                                                  name | linking | request |
+-----------+-------+-------------------------------------------------------+---------+---------+
| 10.1.1.46 | 57431 | 3DE4 Radial - Fisheye, Stereographic, Degree 8 [Plugin] |  plugin |      ok |
| 10.1.1.47 | 57431 | 3DE4 Radial - Fisheye, Stereographic, Degree 8 [Plugin] |  plugin |      ok |
| 10.1.1.50 | 57431 | 3DE4 Radial - Fisheye, Stereographic, Degree 8 [Plugin] |  plugin |      ok |
+-----------+-------+-------------------------------------------------------+---------+---------+
|                                 number of entries listed here: 3                          |
+-------------------------------------------------------------------------------------------+
```

See also

- 4.3 **select...from** *properties*
- 4.3 **select...from** *lens_models*

4.9 UI-Lock related queries

4.9.1 `lock`-*Syntax*

Form

- **`lock name`** *`name`* [**`contact`** *`contact`*]? [**`admin_host_a`** *`host_a`*]? [**`admin_host_b`** *`host_b`*]? [**`rlock`** {**`none`**|**`some`**|**`all`**}]?
- **`lock preset`** *`path_to_preset_file`*
- **`lock`**

Description

By this query the network is locked so that it can be administrated from only one host (or two hosts if specified). This affects access by means of the commandline tool `mhp_admin` as well as the browser interface. When the network is locked, 3DE4-clients can still connect, run computations and modify the number of available threads on their own machine through 3DE4's GUI. In the first form as shown above you provide a human readable name or nickname and optionally a string which expresses ways to contact you (like e.g. `phone 7531`). *`host_a`* is the host you want to restrict the network's access to. The default is the machine from which you send the query, so in many cases you can simply leave it out. *`host_b`* is a secondary administration host. You way want to pass the ip-address of a notebook reserved for administration purposes in your organization, whatever is convenient for you. The default is that there ist no secondary host for administration. Parameter **`rlock`** (for "read-lock") specifies that a certain set of pages can still be examined through the browser interface. The default is that all pages are locked. We currently recommend to leave this parameter out and rely on the default value.

The second form reads the parameters from the preset file passed, which has the following form:

```
<lock>
<!-- recommended -->
        <admin_name>GrumpyZed</admin_name>
<!-- optional -->
        <admin_contact>phone 7531</admin_contact>
<!-- optional. default is the host you are running mhp_admin at. -->
        <admin_host_a></admin_host_a>
<!-- optional, element should exist but may be empty -->
        <admin_host_b></admin_host_b>
<!-- Future use -->
        <permissions>
<!-- Readlock specification. Possible values are: all some none -->
                <rlock>all</rlock>
        </permissions>
</lock>
```

The third form (without parameters) reads the file `~/.mhp_admin/uilock.xml`. This is certainly the fastest and easiest way to lock the network, once you have configured the ui-lock file.

Examples

We start with an unlocked network:

```
mhp_admin select lock host hostname from hosts
```

```
arg (['lock', 'host', 'hostname', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+----------+-------------+------------+---------+
|     lock |        host |   hostname |    role |
+----------+-------------+------------+---------+
|    False |   10.1.1.46 |      box01 |   chief |
|    False |   10.1.1.47 |      box02 |   loyal |
|    False |   10.1.1.50 |      box03 |   loyal |
+----------+-------------+------------+---------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

Now we lock it to the machine the tool is running on

```
mhp_admin lock name "uwe" contact "forbidden forest"
```

The tool will ask for a password:

```
Enter new password: *****
```

and eventually respond

```
+-------------------------------------+
|                lock                 |
+-------------------+-----------------+
|              prop |           value |
+-------------------+-----------------+
|   request sent to | 10.1.1.46:57431 |
|            status |              ok |
| num_hosts_success |               3 |
```

```
|  num_hosts_failed |               0 |
|      admin host a |        10.1.1.44 |
|      admin host b |          0.0.0.0 |
+-------------------+-----------------+
```

Let us verify that locking has been successful:

```
mhp_admin select lock host hostname from hosts
```

The result is what we can expect.

```
arg (['lock', 'host', 'hostname', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+---------+--------------+------------+---------+
|    lock |         host |   hostname |    role |
+---------+--------------+------------+---------+
|    True |    10.1.1.46 |      box01 |   chief |
|    True |    10.1.1.47 |      box02 |   loyal |
|    True |    10.1.1.50 |      box03 |   loyal |
+---------+--------------+------------+---------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

See also

- 4.9.2 **unlock**
- 4.9.3 **relock**
- 4.11 **show** *lock*

4.9.2 unlock-*Syntax*

Form

- **unlock**
- **unlock** *host_a*

Description

The first form works if the machine running the tool is the same as the admin host used for locking. The second form is used to unlock the network from any host in the local area network.

Examples

If we assume the network is still locked from our example in the previous section, we simply call:

```
mhp_admin unlock
```

enter the password and get:

```
+-------------------------------------+
|               unlock                |
+-------------------+-----------------+
|              prop |           value |
+-------------------+-----------------+
|   request sent to | 10.1.1.46:57431 |
|            status |              ok |
| num_hosts_success |               3 |
|  num_hosts_failed |               0 |
+-------------------+-----------------+
```

and verify the network is now unlocked:

```
mhp_admin select lock host hostname from hosts
```

```
arg (['lock', 'host', 'hostname', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+----------+-------------+------------+---------+
|     lock |        host |   hostname |    role |
+----------+-------------+------------+---------+
|    False |   10.1.1.46 |      box01 |   chief |
|    False |   10.1.1.47 |      box02 |   loyal |
|    False |   10.1.1.50 |      box03 |   loyal |
+----------+-------------+------------+---------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

See also

- 4.9.1 **lock**

- 4.9.3 **relock**
- 4.11 **show** *lock*

4.9.3 relock-*Syntax*

Form

- **relock** *host*? **to** *host_a host_b*?

Description

Given a locked MHP-network, by this query you change the lock state from *host* (or the host mhp_admin is running on) to a new admin host *host_a* and optionally a secondary admin host *host_b*. This requires to enter the current password at prompt. The MHP-network remains locked.

Examples

For the following example we consider a network of three nodes 10.1.1.{46,47,50} and two hosts 10.1.1.{44,103} for administration. Given a network

```
mhp_admin select lock host hostname role from hosts
```

```
arg (['lock', 'host', 'hostname', 'role'],)
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.46:57431    |
+---------+--------------+------------+---------+
|    lock |         host |   hostname |    role |
+---------+--------------+------------+---------+
|    True |    10.1.1.46 |      box01 |   chief |
|    True |    10.1.1.47 |      box02 |   loyal |
|    True |    10.1.1.50 |      box03 |   loyal |
+---------+--------------+------------+---------+
| number of nodes listed here - good: 3, bad: 0 |
+-----------------------------------------------+
```

locked to admin host 10.1.1.44

```
mhp_admin show lock
```

```
+---------------------------+
|       current lock        |
+---------------+-----------+
|          prop |     value |
+---------------+-----------+
|        status |        ok |
|     is_locked |      true |
|    admin_name |       uwe |
| admin_contact |  phn 4321 |
|  admin_host_a | 10.1.1.44 |
|  admin_host_b |   0.0.0.0 |
|   permissions |       all |
+---------------+-----------+
```

we move to a different machine by

```
mhp_admin relock to 10.1.1.103
```

The lock state is now

```
mhp_admin show lock
```

```
+----------------------------+
|        current lock        |
+---------------+------------+
|          prop |      value |
+---------------+------------+
|        status |         ok |
|     is_locked |       true |
|    admin_name |        uwe |
| admin_contact |   phn 4321 |
|  admin_host_a | 10.1.1.103 |
|  admin_host_b |    0.0.0.0 |
|   permissions |        all |
+---------------+------------+
```

Any request from the original admin host is blocked now:

```
mhp_admin select all from hosts
```

```
+-----------------------------------------+
```

```
| request http://10.1.1.46:57431/hosts.xml |
+----------------------+-------------------+
|    status of request |    not authorized |
+----------------------+-------------------+
|      chief host:port |   10.1.1.46:57431 |
+----------------------+-------------------+
```

We can however move control back to our original host from anywhere in the local area network by

```
mhp_admin relock 10.1.1.103 to 10.1.1.44
```

so that the lock state becomes

```
mhp_admin show lock
```

```
+---------------------------+
|       current lock        |
+---------------+-----------+
|          prop |     value |
+---------------+-----------+
|        status |        ok |
|     is_locked |      true |
|    admin_name |       uwe |
| admin_contact |    office |
|  admin_host_a | 10.1.1.44 |
|  admin_host_b |   0.0.0.0 |
|   permissions |       all |
+---------------+-----------+
```

Caveat

There is a situation in which relocking as in the previous example is not possible any longer. Let us assume there is a second network (locked or unlocked) with chief `10.1.1.24`. `mhp_admin` is attached to network `10.1.1.46` and admin host is `10.1.1.44` as in the example. Then by

```
mhp_admin relock to 10.1.1.103
mhp_admin at 10.1.1.24
```

you lock yourself out. Working on `10.1.1.44` you cannot go back to network `10.1.1.46` because it is locked, and you cannot unlock or re-lock because `mhp_admin` is attached to a different network. The only ways to unlock or relock is by doing this from host `10.1.1.103` or by editing file `~/.mhp_admin/current.xml` on `10.1.1.44` by hand, like so:

```
<current>
        <host>10.1.1.46</host>
        <port>57431</port>
        <chief_host>10.1.1.46</chief_host>
        <chief_port>57431</chief_port>
</current>
```

and then

```
mhp_admin relock 10.1.1.03 to 10.1.1.44
```

plus password.

See also

- 4.9.1 **`lock`**
- 4.9.2 **`unlock`**
- 4.11 **`show`** *`lock`*

4.10 Tags and views

4.10.1 `create-tag`-*Syntax*

Form

- **`create tag`** [**`for theme`** *`theme`*]? **`if`** *`column function value`* **`bold`**? [*`red green blue`*]?

Description

Create an optical highlight if the table entry of column *`column`* matches value *`value`* with respect to function *`function`*. The function can be `eq` for "equal" or `neq` for "not equal". The highlighted text can appear as bold text and coloured, provided the console running `mhp_admin` allows ANSI terminal colours. `mhp_admin` supports a `dark` and a `light` console color theme. By providing the **`for-theme`**-clause you can assign the tag to one of these themes. If you omit this clause, the tag is assigned to the theme specified in the config file `~/.mhp_admin/con-`

`fig.xml`. The color specification is optional. If you omit this, the color for this tag is inherited from the console (which is equivalent to "no color highlight").

Examples

By the following query table entry `chief` is rendered in the specified color whenever it appears in a column *role*:

```
mhp_admin create tag for theme light if role eq "chief" bold 127 63 0
```

The entire set of tags could look like e.g.:

```
mhp_admin show tags
```

```
+-----------------------------------------------------+
|                tags for theme 'light'               |
+-------------+------+------------+-------+-----------+
|      column | func |      value |  bold |     color |
+-------------+------+------------+-------+-----------+
| launched_as |   eq | standalone | False |   0 127 0 |
| launched_as |   eq | controlled | False |  159 91 0 |
|        role |   eq |      chief |  True |  127 63 0 |
|        role |   eq |      loyal |  True | 91 127 63 |
|        role |   eq |   renegade |  True |   255 0 0 |
|        role |   eq |    unclear |  True | 127 31 31 |
|          rc |   eq |          0 |  True |   0 255 0 |
|          rc |  neq |      not 0 |  True |   255 0 0 |
+-------------+------+------------+-------+-----------+
```

and the list of hosts:

```
mhp_admin select host role from hosts
```

```
+-----------------------------------------------+
|    nodes governed by chief 10.1.1.24:57431    |
+------------------------+----------------------+
|                   host |                 role |
+------------------------+----------------------+
|              10.1.1.24 |                chief |
|              10.1.1.42 |                loyal |
|              10.1.1.44 |                loyal |
|              10.1.1.46 |                loyal |
|              10.1.1.50 |                loyal |
|             10.1.1.103 |             renegade |
|             10.1.1.116 |              unclear |
+------------------------+----------------------+
| number of nodes listed here - good: 5, bad: 2 |
+-----------------------------------------------+
```

which happens to be in a bad state (one node renegade, one node unclear) so that you can see the effect of tags.

See also

- 4.10.2 **drop-tag**-Syntax
- 4.11 **show tags**

4.10.2 drop-tag-*Syntax*

Form

- **drop tag** [**for theme** *theme*]? **if** *column function value*

Description

Remove the tag for theme *theme* where column *column* matches value *value* with respect to function *function*. The function can be `eq` for "equal" or `neq` for "not equal". If the **for-theme**-clause is not specified, the query is applied for the theme given in the config file.

Examples

Using this query is straight forward. Each tag is uniquely determined by *theme*, column *column*, *function* and *value*:

```
mhp_admin drop tag for theme dark if role eq "chief"
```

See also

- 4.10.1 **create-tag**-Syntax
- 4.11 **show tags**

4.10.3 create-view-*Syntax*

Form

- **create view** *view* **as** *query*

Description

This query allows to define a shortcut for **select**-queries. The shortcut is called a **view**, and it is executed y means of the **show**-query (as opposed to languages like SQL, where a view behaves like a "real" table...).

Examples

Create a view named e.g. `uilock` by means of

```
mhp_admin create view uilock as select host hostname is_locked admin_host_a from properties at all
```

and run it with

```
mhp_admin show uilock
```

as a shortcut for

```
mhp_admin select host hostname is_locked admin_host_a from properties at all
```

You can get an overview over the currently defined views by

```
mhp_admin show views
```

which might look like

```
+-------------------------------------------------------------------------+
|                                  views                                  |
+--------+----------------------------------------------------------------+
|   name |                                                          query |
+--------+----------------------------------------------------------------+
|   load |                            select all from load order by host asc |
| uilock | select host hostname is_locked admin_host_a from properties at all |
|  hosts |                   select all from hosts order by role asc host asc |
+--------+----------------------------------------------------------------+
```

See also

- 4.10.4 **drop-view**-Syntax
- 4.11 **show**-Syntax

4.10.4 drop-view-*Syntax*

Form

- **drop view** *view*

Description

Delete the view *view*.

Examples

```
mhp_admin drop view uilock
```

See also

- 4.10.3 **create-view**-Syntax

4.11 show-Syntax

In contrast to the highly polymorphic **select**-queries, **show** represents a simple way of retrieving data. The tool will provide these data as tables but the choice of columns is fixed. Most of the data queried by show refer to the tool itself, but you can also run predefined view queries on the network.

Form

- **show** {*hostbases* | *remote* | *launchers* | *lock* | *tags* | *version* | *views*}
- **show** *view*

Description

show *hostbase* - Show host addresses as given in config file.
show *remote* - Show remote shell commands as given in config file.
show *launchers* - Show MHP-node launchers as given in config file.
show *lock* - Display current lock state of the network.
show *tags* - Show markup tags (bells and whistles)
show *version* - Display version number in machine-readable style.
show *views* - Show user-defined views
show *view* - Run a user-defined query referred to by *view*.

Examples

The queries of the first form are easy to use, just try out. For the second form let us assume we have created views previously:

```
mhp_admin show views
```

```
+-------------------------------------------------------------------+
|                              views                                |
+-------+-----------------------------------------------------------+
|  name |                                                     query |
+-------+-----------------------------------------------------------+
| hosts | select all from hosts order by role asc host asc port asc |
|  proc |                                select all from proc at all |
+-------+-----------------------------------------------------------+
```

Then we can simply write

```
mhp_admin show hosts
mhp_admin show proc
```

instead of repeating the lengthy queries.

4.12 Shortcuts

Since the queries can become quite lengthy, there are several methods to abbreviate them.

- by means of user-defined views as shown in section 4.10.3
- by abbreviating keywords
- by abbreviating ip-addresses

Most (but not all) keywords and table names can be abbreviated by simply typing the first letter and a selection of subsequent letters. We have not done this in the document, because syntax highlighting in our documentation system does not like this... Please consider the following examples:

```
mhp_admin sl all from hosts
```

```
mhp_admin sl all from prop at 10.1.1.24
```

```
mhp_admin sl name from lmod at 10.1.1.50
```

where we type

- `sl` or similar instead of `select`
- `prop` or similar instead of `properties`
- `lmod` or similar instead of `lens_models`

Just leave out letters you don't feel like typing. Ip-addresses can be abbreviated as well. The tool will reconstruct the missing parts from the address representing the current network (see **at**-query). So, for a network governed by chief `10.1.1.24`, instead of typing

```
mhp_admin terminate 10.1.1.46 10.1.1.47 10.1.1.50 10.1.1.103 10.1.1.116
```

you can also type

```
mhp_admin term 1.1.46 1.1.47 1.1.50 1.1.103 1.1.116
```

or

```
mhp_admin term 1.46 1.47 1.50 1.103 1.116
```

or even

```
mhp_admin term 46 47 50 103 116
```

Splitting an MHP-network into two networks reduces from

```
mhp_admin unlink 10.1.1.50
mhp_admin relink 10.1.1.46 10.1.1.47 to 10.1.1.50
```

to e.g.

```
mhp_admin unl 50
mhp_admin rel 46 47 to 50
```

We prefer not to standardize this further, as some shortcuts may be dropped in future versions of the tool to keep the query language well-defined. Please feel free to try this out, but beware of ambiguities from abbreviating too much.

A Style `unicode`

In section 2.2.1.1 we mentioned that the element **<style>** in section **<local>** of the configuration file allows to render tables in unicode characters. For reasons of completeness, here is a table rendered in this style. According to our experience terminals usually render the box glyphs in better quality than browsers, but please try out yourself.

nodes governed by chief 10.1.1.24:57431							
lock	host	hostname	proc	msg	role	up_dhms	version
False	10.1.1.24	guul	daemon	ok	chief	4d 2h	2.0.4
False	10.1.1.42	demon	daemon	ok	loyal	4d 1h	2.0.4
False	10.1.1.47	box02	daemon	ok	loyal	4d 2h	2.0.4
False	10.1.1.50	box03	daemon	ok	loyal	1d 3h	2.0.4
False	10.1.1.46	box01	daemon	ok	loyal	4d 2h	2.0.4
False	10.1.1.52	boxoffice1	daemon	ok	loyal	4d 2h	2.0.4
False	10.1.1.88	mibox	daemon	ok	loyal	4d 1h	2.0.4
False	10.1.1.91	heibox	daemon	ok	loyal	29m 23s	2.0.4
False	10.1.1.87	minimuff.box	daemon	ok	loyal	3d 9h	2.0.4
False	10.1.1.116	applejack	daemon	ok	loyal	3d 9h	2.0.4
False	10.1.1.118	azrael	daemon	ok	loyal	1d 6h	2.0.4
number of nodes listed here - good: 11, bad: 0							

Notes

1.4.2-1: I'm absolutely not an expert for `pip3`, and my experience up to now is - let's say - ~~catastrophic~~ sobering, but try yourself.

Document created with CDML from /server/devel/sdv/privat/uwe/source/tde4mhp_admin/doc/manual/cdml//mhp_admin_manual.xml

© 2015 by Science-D-Visions.