

Using the NMM Registry

Motama GmbH, Saarbruecken, Germany
(<http://www.motama.com>)

April 2010

Copyright (C) 2005-2010
Motama GmbH, Saarbruecken, Germany
<http://www.motama.com>

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being all sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found in the file COPYING.FDL.

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

1. Introduction

NMM uses an application called *serverregistry* for managing hardware and software components available on a system, such as sound cards or software decoders. The *serverregistry* enables applications to search and reserve available plug-ins. Furthermore, the *serverregistry* implements some additional features like transactions or automatic releasing of nodes that were not released by an application.

The information about available plug-ins is stored in a configuration file. This information is used by NMM applications to load required plug-ins on demand. If you start an NMM application for the first time, this information is created automatically, but you must recreate it whenever you change your hardware

configuration or update to another NMM version. In this case, you need to start the *serverregistry* with the following option.

```
serverregistry -s
```

The *serverregistry* then detects all installed plug-ins and prints out a list of plug-ins that are available or not available.

```
Create config file with plugin information ...
Loading plugins...
AACAudioDecodeNode          available
AVDemuxNode                  available
...

Config file successfully written.
```

For an application to be able to create a distributed flow graph, the NMM application called *serverregistry* needs to be running on each participating host. For purely locally operating applications this is not required. Then, a server registry is automatically running within the application itself but not accessible from remote hosts.

To start the *serverregistry* for device management, you must run the command

```
serverregistry
```

The *serverregistry* starts listening on port 22801 to accept incoming connections from applications. When the *serverregistry* is started, the message

```
serverregistry successfully started!
```

appears, and the *serverregistry* accepts some commands from the console that are described in the following section. To stop the *serverregistry* enter 'q' or 'exit' and press enter. You can query all possible command line options of the *serverregistry* application, by running

```
serverregistry -h
```

2. Additional Commands

After starting the *serverregistry*, it accepts several commands from the console after startup that are described in this section.

- **commands** : Prints the commands available and their corresponding descriptions.
- **help** *parameter* : Prints the description for a specific command. Expects the name of a command as parameter
- **exit** or **q** : Exits the *serverregistry*.
- **reset** : Resets the internal parameters of the *serverregistry* and releases all reserved nodes.
- **error-stream** *parameter* : Enables/disables the error stream. Expected parameter 'on', 'off'.
- **warning-stream** *parameter* : Enables/disables the warning stream. Expected parameter 'on', 'off'.
- **debug-stream** *parameter* : Enables/disables the debug stream. Expected parameter 'on', 'off'.
- **message-stream** *parameter* : Enables/disables the message stream. Expected parameter 'on', 'off'.
- **streams** *parameter* : Enables/disables all output-message streams. Expected parameter 'on', 'off'.
- **nodeinfo** *parameter* : Displays information about the node given as parameter, such as the names of input and output streams and supported formats.