MediaManager user guide

John-Paul Stanford <dev@stanwood.org.uk>

MediaManager user guide

by John-Paul Stanford

This document describes usage of the application MediaManager version 2.2

Table of Contents

List of Tables

List of Examples

Chapter 1. Introduction

MediaManager is a application and a API which can be used to retrieve TV show and movie meta data from Internet sources. This information is then stored locally and can be used to perform actions upon the file files, such as renaming the media files based on meta data.

Features

- Provides a CLI tool that renames media files with the correct names.
- Fetches TV Show and Film meta data from the Internet.
- Multiple sources of media information including XBMC meta data parsers.
- Cache media information locally in stores, including a XML store.
- Run actions on the media, such as renaming it based on media information
- Provides a action for executing system commands on media files
- Provides a action creating video podcasts of media files
- Provides a API for accessing the media information.
- Provides a store that writes Sapphire XML files.
- Provides a store that saves meta data into MP4/M4V files that iTunes can read.
- Allows plugins to be registers so that custom Sources, Stores and Actions can be added.
- Provides a command line tool to manage XBMC addon scrapers.
- Runs on any platform that supports Java 1.6.

Sources

Sources are the places that MediaManager retrieves media information from. Sources are read only, so it's not possible to write information back too them. The following sources come with MediaManager, however others can be added.

org.stanwood.media.source.xbmc.XBMCSource

This is the main source that MediaManager was designed to use. It makes use of XBMC scrappers to fetch media information. It is capable of reading both TV Show and Film information if the XBMC Scraper supports it.

In order for this source to work, their must be XBMC addon scrapers installed. These can be installed and managed using the command line tool mm-xbmc

Any parameters set on this source are passed through to the XBMC addon scrapers as parameters. In addition this source also supports the following parameters

- `scrapers` a comma separated list of XBMC scraper ID's. These are the ID's of scrappers that are usable. If not given, then all scrappers are considered usable.

Configuration

In addition to the source parameters, their is also a section of the configuration file that holds settings for all the XBMC sources. The XBMCAddon element in the configuration file has the following optional attributes:

- `directory` - The directory one system to store addons. Defaults to "$HOME/.mediaManager/addons".
- `locale` - The locale to use with the addons. Defaults to "en".
- `addonSite` - The site to install addons from. Defaults to "http://mirrors.xbmc.org/addons/dharma".

Example 1.1. XBMC settings example

```
<mediaManager><XBMCAddons directory="$HOME/.mediaManager/addons"locale="en"addonSi
```

org.stanwood.media.source.TagChimpSource

The TagChimpSource Source is film source that fetches film information from the website www.tagchimp.com [http://www.tagchimp.com/]. This also includes chapter information.

org.stanwood.media.source.HybridFilmSource

The HybridFilm Source is a bit different to the other sources. It uses the other sources to find the best information it can about a film. It's main use is to combine the extra information film chapter information from the `TagChimpSource` source with the information in the `XBMCSource`.

This source supports the following parameters:

- `xbmcSourceId` Id of XBMC source to use, if parameter is not specified, then the default is used.

Stores

Stores are similar too sources, except that they are also writable. Once information has been retrieved from a source, it is written info a store. Next time the information is needed, it can be retrieved from the store. This makes repeatedly retrieving information a lot faster.

Stores can have parameters. These parameters are entered via the configuration file. The stores will validated their parameters when the store is access and report back any problems that are found.

Thee following stores that come with MediaManager. Each of these has different properties. Some of these stores can also be used by other applications such as media centre's to display media meta data.

org.stanwood.media.store.memory.MemoryStore

This store is used to store the media information in memory. This allows the tool to reuse the media information without having to fetch it from other stores or sources (which would be slower). This information will be lost once the application exits.

org.stanwood.media.store.db.DatabaseStore

This is a store used to store TV show and film information in a database. The database can be remote or local. This store is also used when searching for media details. The database connection details should be specified in the the section called “Resources” section of the configuration.

The store needs a empty database to be created and the user specified in the resource must have access to it. Upon first connection, the store will create the database tables.

Example 1.2. Local MySQL database setup

The following commands in the mysql console will setup the database when connecting locally.

```
mysql> CREATE DATABASE mediamanager;mysql> GRANT ALL PRIVILEGES ON mediamanager.*
```

Using the database created above, the following configuration would connect the store to the database:

```
<mediaManager><mediaDirectory directory="/media/films" mode="FILM"><stores><store
```

This store has the following parameters:

- resourceId - The ID of the resource that specifies the connection information. This is a required parameter.

org.stanwood.media.store.db.FileDatabaseStore

This is a store used to store TV show and film information in a database file. File file is stored withing the MediaManager configuration directory. This store is also used when searching for media details. This is the default store when no stores are specified for a media directory.

This store has no parameters

org.stanwood.media.store.xmlstore.XMLStore2

This store is used to store the TV show and film information in a XML file called '.mediaManager-xmlStore.xml'. This file will be located in the root of the media directory and can store multiple films/ tv shows. This store can also be used when searching for TV show and film ID's.

org.stanwood.media.store.SapphireStore

This is a write only store that is used to store information in a format that can be used by the Sapphire [http://appletv.nanopi.net/] frontrow plugin. The details of the XML format can be found here [http:// appletv.nanopi.net/manual/overriding-metadata/].

Every time episode or film information is fetched from a source, a XML file is written with the same name as the episode or film file (except the extension is changed too .xml).

This store has the optional parameter "PreferredCertificationCounrty". If this is set, then when fetching the rating, this country in the parameter's rating is used. If the parameter is not set or the country can't be found, then the first rating is used.

It has the following parameters:

- `PreferredCertificationCounrty` The preferred counties rating to use.

org.stanwood.media.store.mp4.MP4ITunesStore

This store is used to store Film/TV show information in .mp4/.m4v files used by iTunes. This allows iTunes to use the meta data and see the files complete with their meta data.

In order to function, this store uses the command line tools provided by the AtomicParsley application. Their are different forks of this application on the Internet. The most feature rich version I've found is at AtomicParsley [https://bitbucket.org/shield007/atomicparsley]. MediaManager uses this one to add atoms that some of the other versions can't. The application must be installed on the PATH, or pointed to by the optional store parameters.

MediaManager should be able to find these tools if it was installed using the installer. However if it was installed via Linux packages then it will look for them on the system path. It's also possible to tell it where to look for them. See the the section called “Native applications and libraries” chapter or the optional parameters of the store for more information.

If using a version of AtomicParsley that does not support the setting of all fields that this store can handle, then a warning will be printed. A version with the above link that fully supports this store can downloaded from the MediaManager website or installed via the installer.

This store has following optional parameters:

- `atomicparsley` The path to the AtomicParsley command

org.stanwood.media.store.mp4.itunes.RemoteMacOSXItunesStore

This store is used to inform iTunes of file changes in a media directory. It does this by talking to a remote server running on the same machine as iTunes. The details of the server can be found at iTunes Remote Server [http://code.google.com/p/itunes-remote-control-server/].

The optional parameter file-separator can be used when MediaManager is running on a different operating system to the remote client. So for example if MediaManager is on a Linux OS and the remote server is on a windows OS, then the file separator should be set to \. See the page Regex [http://en.wikipedia.org/wiki/Regular_expression] for more information on regular expression syntax.

The search and replace optional parameters can be used to the media directory is access at a different location on the iTunes server machine to the machine that MediaManager is running on.

This store has following parameters:

- `hostname` The path to the AtomicParsley command.
- `port` Optional parameter giving port number of the server, defaults to 7000.
- `username` Required parameter giving name of user used to log into the server.
- `password` Required parameter giving password of user used to log into the server.
- `search-pattern` Optional parameter that must be used with search-replace. This parameter is used to perform a regular expression search and replace on the file paths. This parameter is used to set the pattern.
- `search-replace` Optional parameter that must be used with search-replace. This parameter is used to perform a regular expression search and replace on the file paths. This parameter is used to set the replacement text.
- `file-separator` Optional parameter that is used to set the file separator used in file names sent to the server.

Actions

Actions are tasks that are to be performed upon media files by the application mm-manager. The following actions are provided:

org.stanwood.media.actions.command.ExecuteSystemCommandA

This action is used execute a system command upon media files and directories

This action supports the following parameters, which are all optional:

- `commandOnFile` A command to execute on finding acceptable media files.
- `commandOnDirectory` A command to execute on finding acceptable directories within the media directory.
- `extensions` A comma separated list of media file extensions to accept.
- `newFile` If this command creates a new file, then the name should be in this parameter.
- `deletedFile` If this command deletes a new file, then the name should be in this parameter.
- `abortIfFileExists` The name of a file, that if it exists, then this action will not perform.

Parameters can also have variable in them. These can be any of the pattern variables, as well as the following special variables:

- `$NEWFILE` The value of the 'newFile' parameter.
- `$DELETEDFILE` The value of the 'deletedFile' parameter.
- `$MEDIAFILE_NAME` The name part of the current media file been processed. So after the last file separator, until it finds the extension.
- `$MEDIAFILE_EXT` The extension of the current media file been processed.
- `$MEDIAFILE_DIR` The directory the current media file is in.
- `$MEDIAFILE` The full path of the current media file.
- `$HOME` The current users home directory.

org.stanwood.media.actions.podcast.PodCastAction

This action is used create a pod cast of media that it finds. It will add order the most recent media files by the date they were last modified.

This action supports the following parameters:

- `mediaDirURL` - This is a required parameter that specifies the URL used to find the root media directory.
- `fileLocation` - This is a required parameter that specifies the location of the RSS feed relative to the root of the media directory. It can contain standard rename patterns with the value.
- `numberEntries` - The maximum number of entries in the feed. The default if not set is unlimited.

- `extensions` - A comma separated list of media file extensions to accept.
- `restrictPattern` - This can be used to restrict the media files. It can contain standard rename patterns with the value.
- `feedTitle` - Used to give a title to the RSS feed. It can contain standard rename patterns with the value.
- `feedDescription` - Used to give a description to the RSS feed. It can contain standard rename patterns with the value.

Parameters can also have variable in them. These can be any of the following special variables:

- `$MEDIAFILE_NAME` The name part of the current media file been processed. So after the last file separator, until it finds the extension.
- `$MEDIAFILE_EXT` The extension of the current media file been processed.
- `$MEDIAFILE_DIR` The directory the current media file is in.
- `$MEDIAFILE` The full path of the current media file.
- `$HOME` The current users home directory.

org.stanwood.media.actions.rename.RenameAction

This action is used to rename media files in a media directory based on a pattern found in the configuration file. The action can also move the file to different directory if the pattern has directories in it.

This action supports the following parameters:

- `pruneEmptyFolders` - If true, then after renaming, empty folders will be deleted.

Chapter 2. Installation

Their are a few different ways to install MediaManager. The different distributions can be found at *Downloads* [http://code.google.com/p/tv-and-movies-meta-data-fetcher/downloads/list]

All of the distributions will require a Java 1.6 compatible JRE.

1. Linux Binary Distribution Packages

Their are several Linux distribution packages that can be found in the downloads section of the website. If your distribution uses RPM packages, then following these instructions.

1. Pick the packages for your distribution
2. Download them to a directory
3. Change to that directory from the console
4. Log in as root and run the command **rpm -Uvh *.rpm**

This will install the application and a scripts too launch it.

2. Installer

Their is a generic Java installer that will run on any platform. Execute the following command upon the installer jar file to start the installer:
`Java -jar MediaManager-2.2-install.jar` [options]
This will prompt you for the location to install the application and create start menu links to the documentation.

Their are also platform specific installers that can be launched easier on some platforms. Here are the names of the installers and the platforms they run on:

- `MediaManager-2.2-install.sh` - Linux installer
- `MediaManager-2.2-install.windows.zip` - Windows installer
- `MediaManager-2.2-install.jar` - Any Platform installer (including Mac OSX)

More installation guides can be found on the MediaManager Wiki [http://code.google.com/p/tv-and-movies-meta-data-fetcher/w/list]

Chapter 3. CLI Usage

mm-manager

The mm-manager command is used to managed a media directory. It reads the configuration file to work out which sources, stores and actions are to be used with media directory. Then the actions are performed on the media directory.
`mm-manager` [-v] [-h] [-c *`config file`*] [-l *`INFO|DEBUG|log4j configuration file`*] [-d *`media directory`*] [-t] [-u]

The command has the following options:

- `-v, --version` Display the version
- `-h, --help` Show the help message
- `-d, --dir` A required option give the location of the media directory to manage
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.
- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.
- `-t, --test` Enable test mode that cause no changes to be written to the filesystem.
- `-u, --noupdate` If option is given, the XBMC addon scrapers are not updated.

mm-copy-store-to-store

The mm-copy-store-to-store command is used to copy media file information from one store to another. This can be useful for migrating media information if add store.
`mm-copy-store-to-store` [-c *`config file`*] [-d *`directory`*] [-f *`from store ID`*] [-h] [-l *`INFO|DEBUG|log4j configuration file`*] [-o *`to store ID`*] [-t] [-u] [-v] [<media files.../directory...>]

The command has the following options:

- `-f, --fromStore` The store to read from
- `-v, --version` Display the version
- `-u, --noupdate` If option is given, the XBMC addon scrapers are not updated.
- `-d, --dir` The media directory been used.
- `-t, --test` Enable test mode that cause no changes to be written to the file system.
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.
- `-o, --toStore` The store to copy to

- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.
- `-h, --help` Show the help message

Example 3.1. Usage examples of mm-copy-store-to-store

Example of copying a single files details from one store to another

```
mm-copy-store-to-store --dir /mounts/TVShows --fromStore org.stanwood.media.store.
```

Example of copying the details of all media files in a media directory from one store to another

```
mm-copy-store-to-store --dir /mounts/TVShows --fromStore org.stanwood.media.store.
```

mm-move-into-media-directory

The mm-move-into-media-directory command is used to move media files into a directory. It then uses the sources and stores with the media file and performs the actions on it. The media files can be either files or directories.
`mm-move-into-media-directory` [-v] [-h] [-c *`config file`*] [-l *`INFO|DEBUG|log4j configuration file`*] [-d *`media directory`*] [-t] [-u] [<media file/directory...>]

The command has the following options:

- `-v, --version` Display the version
- `-h, --help` Show the help message
- `-d, --dir` A required option give the location of the media directory to manage
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.
- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.
- `-t, --test` Enable test mode that cause no changes to be written to the file system.
- `-u, --noupdate` If option is given, the XBMC addon scrapers are not updated.

mm-media-file-info

The mm-media-file-info command is used to list the Apple atoms in a MP4/M4V media file which is given as a argument. It can be used to check the meta data that iTunes will see.
`mm-media-file-info` [-v] [-h] [-c *`config file`*] [-l *`INFO|DEBUG|log4j configuration file`*] [<media file>]

The command has the following options:

- `-v, --version` Display the version
- `-h, --help` Show the help message
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.

- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.

mm-xbmc

The mm-xbmc command is used to manage XBMC addons. It has the sub commands and arguments listed below.

```
usage: mm-xbmc [--global-options] <command> [--command-options] [argu-
ments]
```

Global options:

- `-v, --version` Display the version
- `-h, --help` Show the help message
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.
- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.

Commands:

- `list` - Lists the installed XBMC addons
- `update` - Update the installed XBMC addons to the latest versions
- `install` - Install a new XBMC addon
- `remove` - Install a new XBMC addon

mm-print-db-schema

The mm-print-db-schema command is used to print the database schema for creating an intimal database. A hibernate dialect must be passed to it, see ??? for more information on dialects.
`mm-print-db-schema` [-v] [-h] [-d *`dialect`*] [-c *`config file`*] [-l *`INFO|DEBUG|log4j configuration file`*]

The command has the following options:

- `-v, --version` Display the version
- `-h, --help` Show the help message
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.
- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.
- `-d, --dialect` The database dialect.

mm-import-media

The mm-import-media command is used to check for new media in watched directories. If it find media, it attempts to lookup the information for the files and move them to the correct media directory. See the section called “Watch Directories” for configuration details.
`mm-import-media` [-v] [-h] [-a] [-d] [-e] [-t] [-u] [-c *`config file`*] [-l *`INFO|DEBUG|log4j configuration file`*]

The command has the following options:

- `-v, --version` Display the version
- `-h, --help` Show the help message
- `-c, --config_file` The location of the configuration file. If option is not given then it will load from default locations.
- `-l, --log_config` The log configuration mode. Either INFO, or DEBUG for the built in configurations, or a file name of a log4j configuration file.
- `--dontUseDefaults, -d` Don't use default media directories.
- `--noupdate, -u` If this option is present, then the XBMC addons won't be updated
- `--deleteNonMedia, -e` Delete files are that are not media files (use with care)
- `--test, -t` If this option is present, then no changes are performed.
- `--actions, -a` Execute actions on new media files

Chapter 4. Configuration

The applications and the API make use of a XML configuration file. This stores information about the media directories and how they should be managed.

The applications have CLI options that can be used too tell it which configuration file too use. If this option is not present, then it will look for the file at the location `/etc/mediamanager-conf.xml` and `$HOME/.mediaManager/mediamanager-conf.xml`. If these can't be found, then a default configuration file will be created at `$HOME/.mediaManager/mediamanager-conf.xml`.

Media directories

MediaManager is used to manage media in media directories, this means you need to tell it about the media directories. Their is a mediaDirectory entity in the configuration file used to do this. It has the following attributes:

- `directory` - The location of the media directory.
- `mode` - The type of media. . See the section called “Mode”.
- `pattern` - The rename pattern. See the section called “Patterns”.
- `ignoreSeen` - option attribute, if set to true then once a file is seen it will not be processed again

Mode

Media directories can operate in different modes. This effects how the media is handled and which pattern tokens can be used. The currently supported modes are:

- `TV_SHOW` - Media files are TV episodes
- `FILM` - Media files are films

Patterns

A pattern can be associated with a media directory. This is used by actions like the Rename action to rename file media files and change the directory structure.

The following list contains the meaning for each token:

- `%d` - If HighDef, replaced with "HD"
- `%e` - episode number
- `%f` - last episode number of the episode contains multiple episodes
- `%h` - show Id
- `%i` - the show or film image URL
- `%n` - show name

- `%p` - part number
- `%s` - season number
- `%t` - episode title or film title
- `%u` - the show or film short summary
- `%w` - If Wide screen, replaced with "WS"
- `%x` - extension (avi, mkv....)
- `%y` - year
- `%%` - add a % char

The patterns also support the syntax `"{ Part %p }"`. The braces means the contents inside are optional. So in this case if the media does not have a part number, then it would be added to the filename.

Example 4.1. Examples:

The following options show what happens when they are used to rename the 4th episode of the 2nd season of the show heroes.

- "S%s E%e - %t.%x" = "S2 E04 - The Kindness of Strangers.avi"
- "%sx%e.%n.%t.%x" = "2x04.Heroes.The Kindness of Strangers.avi"
- "%sx%e.%h.%t.%x" = "2x04.17552.The Kindness of Strangers.avi"

Watch Directories

MediaManager is able to monitor file system directories for new media. When it finds it, it can be processed and moved into media directories. Use the command the section called “mm-import-media” to check for new media. The following example shows how to setup a file system directory for monitoring.

Example 4.2. Watch Directory

```
<mediaManager><watchDirectory directory="/mounts/newMedia"/></mediaManager>
```

Ignore Patterns

MediaManager can be configured to ignore media files that match a regular expressions. The following example shows how to configure ignore patterns that cause files to be not be processed.

Example 4.3. Ignore patterns

```
<mediaManager><mediaDirectory directory="/media/films" mode="FILM" pattern="%t{ Pa
```

Strip patterns

Quite often before a media file is renamed, it can contain tokens to indicate where it came from and the type of media. MediaManager makes use of these when looking up the media information and has a default list of know strip patterns. It's possible to configure different strip patterns for each media directory which will override the default patterns in use. The patterns are regular expressions.

Example 4.4. Strip patterns

```
<mediaManager><mediaDirectory directory="/media/films" mode="FILM" pattern="%t{ Pa
```

Plugins

It is possible to extend MediaManager with plugins. Plugins are capable of adding new Sources, Stores and Actions. Plugins must extend the correct interfaces like the internal Sources, Stores and Actions. They must then be packaged into a .jar file and imported into MediaManager. The following example shows the configuration file entries needed to tell MediaManager about a plugin.

Example 4.5. Registering plugins

```
<mediaManager><plugins><plugin jar="$HOME/.mediaManager/plguins/myplugin.jar" clas
```

Native applications and libraries

Some parts of MediaManager depend on native applications/libraries to function. Usually MediaManager runs on any platform that supports Java 1.6. However these parts need native applications, so they must either be installed on the system or MediaManager must be told how to find them. If they can't be found, then parts of MediaManager that use then will be disabled.

If MediaManager was installed via the installer, then the native folder will have been installed already and it should find then. If installing from a Linux package, then their is a chance that native applications and libraries won't be found. MediaManager will fist look for them in a folder pointed to by configuration, then an environment variable MM_NATIVE_DIR and lastly on the system path.

If they are not install, then they can be downloaded from the media manager website at http://code.google.com/p/tv-and-movies-meta-data-fetcher/downloads/detail?name=MediaManager-2.2-native.zip&can=2&q=. The native folder should be unzipped from the downloaded file and pointed to by the configuration. See the section the section called “Global Settings” [14].

Global Settings

Their are some global settings which can be configured. Below is a list of settings:

- configDirectory - The location of the local configuration directory
- native - A directory which contains the native applications/libraries used by MediaManager

The native setting is used to specify the location of a directory containing the native applications and libraries that MediaManager depends on. See the the section called “Native applications and libraries” section for mare information.

Example 4.6. Global settings

```
<mediaManager><global><configDirectory>$HOME/.mediaManager</configDirectory><nativ
```

Scripting

MediaManager can be configured to use scripts. These scripts can be in any langauge supported by Java. For example jruby or jython. In order for this to work the libraries for the language must be on the classpath before lauching MediaManager. The examples in this guide are usally done with jruby.

Scripts can contain functions that are called when certian events occur. These methods have access to globals variables allowed by the script engine. Here is a list of parameters and varaibles.

Table 4.1. Variables

Name	**Description**
log	The logger, used to log output. (Commons logging logger)

Table 4.2. Functions

Name	**Description**
onEventPreMediaImport(mediaDirectory)	Called before media is imported from a media directory
onEventPostMediaImport(mediaDirectory)	Called after media is imported from a media directory
onEventPreManageMedia(mediaDirectory)	Called before a media directory is mananged.
onEventPostManageMedia(mediaDirectory)	Called after a media directory is mananged.

Example 4.7. Configuration

```
<mediaManager><scripts><file language="ruby" location="/a/test/script.rb"/></scrip
```

Example 4.8. Example ruby script

```
def onEventPreMediaImport(watchDirectory)$log.info("onEventPreMediaImport(#{watchD
```

Resources

Resources can be configured in the resources section of the database. Resource information describes connection information to systems resources such as a database. The database resource as the following options:

- id - Used to identify the resource to other parts of the configuration.
- url - The JDBC URL to the database. The URL syntax depends on the database been connected to, but typical contains the hostname and database name.
- dialect - The is the SQL dialect that the hibernate database library should use. Below are some examples:
  - MySQL - org.hibernate.dialect.MySQLDialect
  - HSQLDB - org.hibernate.dialect.HSQLDialect
  - PostgreSQL - org.hibernate.dialect.PostgreSQLDialect
  - Microsoft SQL Server - org.hibernate.dialect.SQLServerDialect
  - Oracle - org.hibernate.dialect.OracleDialect

- username - The name of the user used to access the database
- password - The password of the user used to access the database
- schemaCheck - Optional parameter that controls how the database schema should be checked when connecting to the database. Possible values are "validate" and "none". This defaults to validate which causes the database schema to be validated. On some database connections, the validation reports problems when their are none. So setting this parameter to none, will cause no schema checks to be done.

Example 4.9. MySQL database resource

```
<mediaManager><resources><databaseResource id="mainDB"><url>jdbc:mysql://localhost
```

Example 4.10. PostgreSQL database resource

```
<mediaManager><resources><databaseResource id="mainDB"><url>jdbc:postgresql://loca
```

Seen Database

The seen database is used to store a list of files that have been seen before. This also records the time the file was last modified so that if it's updated the changes will be noticed. Media directories can be set to ignore seen media files. This seeds up the processing of media files. By default the seen database is stored in a file in the configuration directory. It's also possible to store the seen database in a actual database. This is done be setting up a database resource and configuring the seen database to use it.

Example 4.11. Storing the seen database within a database resource

This will set the seen database to use the database resource "mainDB".

```
<mediaManager><resources><databaseResource id="mainDB"><url>jdbc:mysql://localhost
```

Examples

Example 4.12. Default Configuration file

This configuration file shows the required options needed to configure a media directory. It uses default sources, stores and actions. Here is what is used by default with a media directory:

- source - the section called “org.stanwood.media.source.xbmc.XBMCSource”
- stores - the section called “org.stanwood.media.store.db.FileDatabaseStore”, the section called “org.stanwood.media.store.mp4.MP4ITunesStore”
- actions - the section called “org.stanwood.media.actions.rename.RenameAction”

The default stores, sources and actions will ensure that the latest meta data is downloaded for the media directory and that the files are renamed according to their meta data. If the file is a m4v/mp4 file, then meta data is inserted into the file.

```
<!-- A configuration file that uses defualt sources, stores and actions --><mediaM
```

Example 4.13. A more complex configuration file

```
<mediaManager><global><!-- Used to define the location where configuration setting
```

Chapter 5. Credits and Licenses

MediaManager

- John-Paul Stanford `<dev@stanwood.org.uk>` - Original Author