



EMC® Watch4net®

Version 6.4u2

Task Scheduler Administration Guide

P/N: pending

Rev : 48507

EMC²

Copyright ©2011 - 2014 EMC Corporation. All rights reserved. Published in the USA.

Published October, 2014

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (<https://support.emc.com>).

Contents

Preface	iii
Overview	iii
Intended Audience	iv
Common Abbreviations and Acronyms	iv
Where to get help	v
1 Installation and Configuration	1
1 Installation	1
2 Configuration	1
3 Runnning the service	3
2 Tasks	5
1 Scheduling	5
2 Max duration	6
3 Actions	6
4 Locks	6
3 Task execution	7
4 Task types	9
1 Script tasks	9
2 Java tasks	10
3 Task sequence	11
5 Add scheduled tasks	13
6 Task description file	15
1 <schedule>	15
2 <action>	16
2.1 Script action	16

2.2	Java action	17
2.3	Task sequence action	17
3	Other elements	18
7	Examples of task file	19
8	Task execution logging	23
9	Command Line Interface	25
1	Global option	25
2	Available commands	26
2.1	help	26
2.2	list	27
2.3	disable	28
2.4	enable	28
2.5	info	29
2.6	start	29
2.7	stop	30
3	Troubleshooting	30
3.1	Logs	30
3.2	Exit Status	30

Preface

Overview

The Task Scheduler is an APG service that can run scheduled tasks. It's intended to replace Cron on Unix-based systems, and the Schedule Tasks on Windows servers for APG-specific tasks. The Task Scheduler runs scheduled tasks that are important to the smooth functioning of APG, such as, refreshing the Frontend-Search index periodically, or performing database maintenance tasks to optimize performance.

The intervals at which tasks run can be adjusted. Tasks are defined by XML-structured files, described by a schema, and have extension `.task`.

The Task Scheduler monitors a specific directory where task files are located. A task is added to the Task Scheduler for execution when its definition file is put in the directory. It is removed when the file is removed.

Modifications on task files are also automatically monitored by the Task Scheduler and its behaviour is updated accordingly.

Intended Audience

This guide is meant to be read by any person who needs to manage schedule tasks installed by APG or design new ones.

Common Abbreviations and Acronyms

<APG>	refers to the APG installation directory
<instance>	refers to the instance name of the Task Scheduler module (typically "Default")

Where to get help

EMC support, product, and licensing information can be obtained as follows:

PRODUCT INFORMATION - For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

<https://support.emc.com>

TECHNICAL SUPPORT - Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

1

Installation and Configuration

1 Installation

The Task Scheduler is installed automatically with a default installation of APG. Otherwise it can be installed and controlled like all other APG modules/services. See the ***Module Manager Guide*** for more information.

The Task Scheduler, through the **Module-Manager**, automatically searches for and schedules any tasks it finds associated to installed modules. Any module installed subsequently has its associated tasks automatically installed and scheduled to run by the Task Scheduler.

2 Configuration

The Task Scheduler configuration file controls where the task files it runs are located, and where it puts its logs and history files.

The file is located at <APG>/Tools/Task-Scheduler/<instance>/conf/config.xml and the default configuration is:

```
<?xml version="1.0" encoding="UTF-8"?>
<scheduler-config xmlns="http://www.watch4net.com/Tools/Scheduler/Config" xmlns:xsi="http://www
.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Tools/Scheduler/Config␣config.xsd␣">
  <task-files-directory>data/task_repository/</task-files-directory>
  <!--Update the scheduler more frequently than a minute to make it more responsive.-->
  <scheduler-update-delay>1000</scheduler-update-delay>
  <history-files-directory>data/history_cache/</history-files-directory>
  <task-logging-directory>logs/</task-logging-directory>
</scheduler-config>
```

The options that can be set in the config.xml are:

- <task-files-directory> The directory where task files are deposited to be run by the Task Scheduler.

In general it is not recommended to change the location of this directory. Typically, the Task Scheduler is installed by default, and any modules that have associated tasks are added to this directory. If you change this directory, previously scheduled tasks will not be run unless they are also moved to the new directory.

- <task-provider-update-delay> Delay in milliseconds between refreshing the monitor tracking file modifications in the task repository. During each refresh, the addition, removal and modification of task files are reported to the Task Scheduler. Reducing the value allows task to be loaded more quickly, but might increase the CPU load. Default value is one second.
- <scheduler-update-delay> Update delay of the Task Scheduler in milliseconds. Updates are the instants when tasks are started. Using lower delay makes the Task Scheduler more responsive, especially when adding immediate tasks, but might increase the CPU load. Typically, the update delay should be between one second and one minute. Default value is one minute.
- <history-files-directory> The directory where history files are stored. History files keep statistics about the execution of tasks. It is generally recommended to access the task history from the Centralized Management.
- <task-logging-directory> The directory where log files are stored. Log files for the Task Scheduler are kept at the root of this directory, whereas task-specific log files are nested here, each in a folder named for the task they pertain to.

- `<max-log-files>` This limits the number of log files kept for each task to the indicated value. Older log files are replaced by new ones as this number is surpassed. Specifying a value of 0 disables the log creation for each task. By default the limit is 10.

Note: The path of the directories specified in the configuration file are relative to the root directory of the Task Scheduler (`<APG>/Tools/Task-Scheduler/<instance>`).

3 Runnning the service

The Task Scheduler runs like all other APG services. See the ***Module Manager Guide*** for details.

2

Tasks

Tasks are run periodically, or once, at a specific date and time or immediately, executing scripts or Java commands.

A task has four parts:

- a schedule indicating when it will be executed.
- the maximum time the task can take to run.
- an action to take upon execution.
- locks that prevent tasks from running simultaneously.

1 Scheduling

There are two types of scheduling: tasks that will be scheduled once, and tasks whose execution is repeated according to the schedule. Tasks scheduled for one execution are executed only once at the scheduled time or immediately, and then removed from the task repository.

Repeated tasks are executed an infinite number of times. The task is executed each time a cron

expression is verified, for example, every minute, every hour, each day at 2am, each Sunday at 3pm, each 29th of February, and so on.

2 Max duration

Tasks contain a maximum execution time. If a task execution takes more time than its max time, it is killed.

3 Actions

There are three kinds of actions.

- Script actions are actions executing a script file (shell scripts on Unix-based systems, command files on Windows systems). These scripts are inherently highly OS dependant.
- Java actions call a specific method of a given Java class, similar to a java main.
- Action sequences contain a combination of these actions. Each of the actions within the sequence is executed one after another.

4 Locks

A task can contain locks. By default, a task can be executed only if no other task currently being executed has a common lock with this task. For example, the task having the locks `database` and `update` cannot execute simultaneously with a task having the locks `database` and `refresh`. Locks can be any string.

Locks restriction can be loosen by specifying the maximum number of tasks holding the same lock to run simultaneously. By default it is one.

Tasks also contain an implicit lock, preventing two instances of the same task from running in parallel.

3

Task execution

Each task executes in its own process.

A task executes according to its schedule as long as it does not conflict with another task according to its lock values. If a task runs into its next scheduled time, or another with the same lock value, the subsequent running of the task will happen immediately after the first instance finishes running. Only one instance of the task can be queued this way, so if three runnings are skipped, for example, only one is run subsequently.

If the Task Scheduler service goes down while the task is running, the task continues to run to completion, but its running is not recorded in the task history or the logs.

A task history is also stored, one history file per task, including information such as the last execution stats, average duration of executions, etc... By default the file is located in the <APG>/Tools/Task-Scheduler/<instance> directory. It is advised to access the task history from a browser in the Centralized Management.

Task types

1 Script tasks

Script tasks execute a shell script. Their definitions are composed of two parts, the script file and associated parameters to use.

Along with the parameters, the Task Scheduler also provides some environment variables that can be retrieved within the script:

- `APG_SCHEDULER_TASK_ID` The ID of the executed task.
- `APG_SCHEDULER_TASK_NAME` The name of the executed task. This is the value for the name attribute in the `<task>` tag of the task description file.
- `APG_SCHEDULER_TASK_RUN_NUMBER` The incremental number for each execution, starting at 1.
- `APG_SCHEDULER_TASK_FILE_PATH` Path to the file describing the task. The path is relative to the working directory.
- `APG_SCHEDULER_TASK_LOG_DIR` Path to the directory where the logs relative to the task executions are stored. The path is relative to the working directory.

- `APG_SCHEDULER_TASK_FORCED_EXECUTION` This property has a value of "true" when the task execution was started manually. Otherwise, when the task is executed on schedule, its value is "false".

The script file must be executable and its location must be given relatively to the Task Scheduler module's root directory (`<APG>/Tools/Task-Scheduler/<instance>`).

The working directory where the script executes can be specified and is always relative to the Task Scheduler module's root directory. By default the script is executed at the Task Scheduler module's root directory. Including a script's exit value (typically 0 for success, other values for failures) is useful as it is then stored in the task history.

Example

Specifying `../../../../bin/manage-module.sh` as script and the parameters `install` and `task-scheduler` will execute the command:

```
<APG>/bin/manage-module.sh install task-scheduler.
```

2 Java tasks

Java tasks execute a static method similar to a Java main from a class given to the task.

The class must provides the method:

```
public static void execute(String[] args);
```

When the task is executed, the `execute` method is called. The parameters given to the task are given to the method through its `args` parameter.

Additionally the class can provide this optional method:

```
public static void terminate();
```

The `terminate` method is called when the task execution is aborted while running, either by the user or because of max time. Some clean up can be done there.

The code executed by the `execute` and `terminate` methods should not call `System.halt(int)` as it prevents to return the right execution status. `System.err` and `System.out` streams are automatically

redirected into the log file relative to the current execution. For more details about log files see section [Task execution logging](#) page 23.

Note: `execute` and `terminate` are defaults for the method names. Other names can be specified for each task, in their description files. For example, `execute` may be replaced by `main` to use the method `public static void main(String[] args)` as entry point. For more details see section [Java action](#) page 17.

To be executed, a task needs:

- the name of the class containing the method to execute (example: `com.watch4net.apg.MyClass`)
- the class-path where all the classes needed for the execution are located

A working directory can also be specified as in script tasks.

Some properties are also given to the executed methods. They can be retrieved using `System.getProperty` and are:

- `com.watch4net.apg.scheduler.task.id` The ID of the executed task.
- `com.watch4net.apg.scheduler.task.name` The name of the executed task.
- `com.watch4net.apg.scheduler.task.run.number` The incremental number for each execution, starting at 1.
- `com.watch4net.apg.scheduler.task.file.path` Path to the file describing the task. The path is relative to the working directory.
- `com.watch4net.apg.scheduler.task.log.dir` Path to the directory where the logs relative to the task executions are stored. The path is relative to the working directory.
- `com.watch4net.apg.scheduler.task.forced.execution` This property has a value of "true" when the task execution was started manually. Otherwise, when the task is executed on schedule, its value "false".

3 Task sequence

A task sequence executes its subtasks one after an other. It is a container of tasks of the type script and/or Java.

Its executions are considered succesfull if all its subtasks complete successfully. If any of the subtasks fails, the whole execution is immediately stopped and is considered failed.

Add scheduled tasks

To add a scheduled task, add a valid task file to the task repository directory.

PROCEDURE

- Compose a scheduled task `.task` file according to the guidelines in this document.
- Place the file in `<APG>/Tools/Task-Scheduler/<instance>/data/task_repository` (by default).

The task is run the next time its cron schedule resolves, or immediately if it is scheduled to be executed only once and there is no cron indicated.

6

Task description file

Task files (with the file extension `.task`) are placed by default in `<APG>/Tools/Task-Scheduler/<instance>/data/task_repository` to be executed by the Task Scheduler.

Tasks are composed of two main parts: when to launch the task (schedule), and the job to execute (action), along with a few other related elements.

Task Description files are validated by an XML schema:
`<APG>/Tools/Task-Scheduler/<instance>/conf/task.xsd`.

1 `<schedule>`

There are two kinds of schedules: `schedule-repeated` and `schedule-once`.

The `schedule-repeated` type requires a cron expression (such as `"* * * * ?"`; see <http://en.wikipedia.org/wiki/Cron> for details). The task is then executed each time the cron ex-

pression is verified (for example: every minute, every Sunday at 8:00, etc).

If the `schedule-once` type is used, the task is launched only once. Its `cron` attribute is optional. If the `cron` is specified, the task is executed the first time the cron expression is verified. If none is specified the task is launched immediately (at the next Task Scheduler update).

Both schedule types have two other optional attributes: `disabled` and `time-zone`.

If `disabled` is `true`, the task is loaded in the Task Scheduler, but never executed. By default `disabled` is `false`.

If the `time zone` attribute is specified, the cron expression is resolved in this particular time zone. The value of the attribute is the id of Java time zone, such as "America/Los_Angeles" (see <http://docs.oracle.com/javase/6/docs/api/java/util/TimeZone.html> for details). By default the cron is resolved on the Task Scheduler time zone.

WARNING: After the execution of a task with `schedule-once` schedule, its description file is removed from the task repository.

2 <action>

There are three kinds of actions: Script actions, Java actions and task sequence actions, each composed of these components.

2.1 Script action

- `<working-directory>` Location where the script is executed, relative to the root directory of the Task Scheduler module. By default the directory is the root of the Task Scheduler module.

For example, to run the script the root directory of APG, `../..` must be indicated for the path.

- `<script>` Path to the script file to execute, relative to the root directory of the Task Scheduler module.
- `<result-values>` A comma separated list of integers, defining the return values that represent success. If a script execution returns a value which does not belong to the list, it is considered as failed. If the tag is missing, 0 is the only value considered as successful. If the list is empty, every return value is considered as successful.

- `<parameters>` Optional, contains a list of parameters to give to the script (1 or more). Inside the `<parameters>` element, each parameter is defined by a separate `<param>` tag, containing the value of the parameter.

2.2 Java action

- `<working-directory>` Same as `<working-directory>` for the script action, but for the Java process.
- `<executing-class>` Full name of the class containing the static method to execute (example: `com.watch4net.apg.MyClass`).
- `<class-path>` Class path used when executing the task. The class path is given as a set of path relative to the working directory. Paths are separated by ":" and for each one the file separator is "/".
- `<java-home>` Location of the Java Runtime Environment (JRE) installation used to execute the task. If not specified the Task Scheduler uses the JRE that comes with APG.
- `<execute-method>` Name of the static method called when the task is executed. This method is the entry point for the task execution. Default is `execute`.
- `<terminate-method>` Name of the static method called when a task execution is aborted. Default is `terminate`.
- `<exit-values>` Similar to script action's `<return-values>` tag, this list of comma separated integers applies when the execute method ends by a call to `System.exit`. When the execute method returns normally, tasks executions are always seen as successful. If the tag is missing, 0 is the only value considered as successful. If the list is empty, every exit value is considered as successful.
- `<jvm-options>` A set of options given to the Java Virtual Machine (JVM) that executes the task. Inside the `<jvm-options>` element, each option is defined by a separate `<option>` tag. Options' names are specified by the attribute name of the `<option>` tag while the value is given by its content. For the complete set of options that can be given to the JVM see: <http://download.oracle.com/javase/6/docs/technotes/tools/windows/java.html>.
- `<parameters>` Parameters to give to the execute method (0 or more). Similar to the parameters of script actions.

2.3 Task sequence action

A list of Script and Java subactions to execute.

3 Other elements

- `<max-duration>` Given in the form `xdxhxmms` (`x` is a number) for day, hours, minute, seconds (optional).
- `<execution-locks>` Set of locks held when the action is executed. Only a given number of actions holding the same lock can run simultaneously. Inside the `<execution-locks>` element, each lock is defined by a separate `<lock>` element, containing the name of the lock. Locks' names can be any string. `<lock>` elements have an optional attribute `max-simultaneous` defining the maximum number of tasks holding the lock that may be executed simultaneously. Default is 1, meaning that if the task is executing, all other task with this lock needs to wait until it finishes. All tasks have their name as an implicit lock, so two instances of the same task cannot run simultaneously.
- `<provider>` Indicates from which APG's module the task come from (if relevant). This element is informational and does not affect the execution of the task.

7

Examples of task file

A simple script task displaying "APG 6.0" every minute.

```
<?xml version="1.0" encoding="UTF-8"?>
<task xmlns="http://www.watch4net.com/Tools/Scheduler/Tasks" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
      xsi:schemaLocation="http://www.watch4net.com/Tools/Scheduler/Tasks task.xsd" name="a_very_
simple_task">
  <!-- task running every minute -->
  <schedule xsi:type="schedule-repeated" cron="*_*_*_*_*_*?"/></schedule>
  <action xsi:type="script-action">
    <script>example/script.sh</script>
    <parameters>
      <param>APG 6.0</param>
    </parameters>
  </action>
</task>
```

The corresponding script file <APG>/Tools/Task-Scheduler/<instance>/example/script.sh:

```
#!/bin/sh
```

```
echo $1
```

A simple Java task displaying "Watch4net APG! Visualize Beyond" each day at 8am.

```
<?xml version="1.0" encoding="UTF-8"?>
<task xmlns="http://www.watch4net.com/Tools/Scheduler/Tasks" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
      xsi:schemaLocation="http://www.watch4net.com/Tools/Scheduler/Tasks_task.xsd" name="2_
minutes_Java_task">
  <!-- task running every day at 8:00 -->
  <schedule xsi:type="schedule-repeated" cron="0_8_*_*_*"></schedule>
  <action xsi:type="java-action">
    <executing-class>example.java.SimpleProcess</executing-class>
    <class-path>.</class-path>
    <parameters>
      <param> Visualize Beyond</param>
    </parameters>
  </action>
</task>
```

The corresponding class source code:

```
package example.java;

/**
 * Class launched by a java task and just printing some text.
 */
public final class SimpleProcess {

  /**
   * Main function that will be called when the task is launched.
   *
   * @param args
   *        array or parameters.
   */
  public static void execute(String[] args) {
    System.out.println("Watch4Net_APG!" + args[0]);
  }

  /**
   * Called when an asynchronous termination happens.
   */
}
```

```

public static void terminate() {
    // nothing to do.
}

/** utility class. */
private SimpleProcess() {
}
}

```

A simple task sequence with a max duration and a lock, executing the two previous tasks in a row every 4 minutes.

```

<?xml version="1.0" encoding="UTF-8"?>
<task xmlns="http://www.watch4net.com/Tools/Scheduler/Tasks" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
    xsi:schemaLocation="http://www.watch4net.com/Tools/Scheduler/Tasks task.xsd" name="
sequence">
    <!-- task running every 4 minutes -->
    <schedule xsi:type="schedule-repeated" cron="*/4*_*_*_*_*?"/></schedule>
    <max-duration>3h25m10s</max-duration>
    <action xsi:type="action-sequence">
        <action-sequence xsi:type="script-action">
            <script>example/script.sh</script>
            <parameters>
                <param>APG 6.0</param>
            </parameters>
        </action-sequence>
        <action-sequence xsi:type="java-action">
            <executing-class>example.java.SimpleProcess</executing-class>
            <class-path>.</class-path>
            <parameters>
                <param> Visualize Beyond</param>
            </parameters>
        </action-sequence>
    </action>

    <execution-locks>
        <lock>database-access</lock>
    </execution-locks>
</task>

```

Task used in APG to recreate data_property_flat table.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Scheduled task for the maintenance of APG database. -->
<!-- Recreate data_property_flat table. -->
<task xmlns="http://www.watch4net.com/Tools/Scheduler/Tasks" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
      xsi:schemaLocation="http://www.watch4net.com/Tools/Scheduler/Tasks_task.xsd" category="
Database" name="recreate-data-property-flat">
  <!-- task running every 6 hours starting at 1h00 each day. -->
  <schedule xsi:type="schedule-repeated" cron="0 1,7,13,19 * * * *"></schedule>
  <action xsi:type="java-action">
    <executing-class>com.watch4net.component.loader.util.ComponentLauncher</executing-class
  >
    <class-path>lib/component-loader.jar</class-path>
    <execute-method>main</execute-method>
    <jvm-options>
      <option name="D">java.util.logging.config.file=conf/logging.properties</option>
      <option name="javaagent:">lib/apg-bootstrap-agent.jar</option>
    </jvm-options>
    <parameters>
      <param>-F</param>
    </parameters>
  </action>

  <!-- prevent other database maintenance tasks to execute simultaneously. -->
  <execution-locks>
    <lock>database-maintenance</lock>
  </execution-locks>
</task>
```

8

Task execution logging

Like most other APG modules, a log file is created under the logs directory where all the logs are stored. The logs directory is located by default at `<APG>/Tools/Task-Scheduler/<instance>/logs`, but can be changed in the Task Scheduler's `config.xml` file.

Logs specific to an individual task's execution are written in specific files. For each task, a folder is created under the logs directory, where its name is the name of the task file.

Under this directory a separate file is created for each task run. The last (or current) task run log file has the name `taskRun-0-<x>.log`, with the number incrementing back in time, so `taskRun-1-<x>.log` will be the second latest, `taskRun-2-<x>.log` the third latest, and so on.

Command Line Interface

A command line tool is available to manage scheduled tasks. This one allows you to execute the same operations as in the user interface.

The script is located in <APG>/bin/. It is named `manage-tasks.sh` for linux and solaris systems, and `manage-tasks.cmd` for windows ones. There is no need to install anything else than the Task Scheduler module. Once the Task Scheduler module is installed, the command line tool is ready to use.

1 Global option

`--url` option allows to connect to a remote machine where a Task Scheduler module is installed. Without this option, the tool will try to connect to the local Task Scheduler module.

Usage: `manage-tasks.sh <command> --url <url>`
Where <url> is built like this:

```
<protocol>://<user name>[:<password>]@<remote host>:<remote port>/<path>
```

Rules behind this option are as followed:

- protocol is mandatory. Allowed protocols are http or https
- user credentials. User name and password must be separated by a colon. Password can be omitted. In this case, you will be prompted to enter the password.
- url of remote Task Scheduler module. It must be a valid one.

Valid URL examples:

- `https://admin:changeme@<remote host>:48443/Tools/Task-Scheduler/<instance>`
- `https://admin@<remote host>:48443/Tools/Task-Scheduler/<instance>`

Note: If used, this option must be the last one of the command.

2 Available commands

Usage for linux and solaris: `manage-tasks.sh <command> [<command args> ...]`

Usage for windows: `manage-tasks.cmd <command> [<command args> ...]`

Note: To keep the documentation light, extension of the `manage-tasks` command is removed for the remaining sections.

2.1 help

The help command gives usage for each command.

```
manage-tasks help [<command>]
```

Where `<command>` is a command name. If `<command>` is omitted, a global help message is displayed.

2.2 list

The list command gives all available scheduled tasks on a server.

```
manage-tasks list [-f <filter>]
```

The output result is a table containing all available scheduled tasks.

```
[root@apg-server]# bin/manage-tasks.sh list
Scheduled Tasks List:
```

Id	Name	Category	Last Result	Current Status	Cron
daily-health-report.task	Daily Health Report-user1	report-scheduling	unknown	idle	0 0 * * *
import-properties.task	import-properties-Default	Database	success	idle	0 * * * *
refresh-topology.task	refresh-topology-Default		unknown	idle	0 22 * * *

Note: Tasks above have been shortened to make easier reading.

-f option allows to filter the output displayed tasks.

Usage: manage-tasks list -f [id|name|cat|category|result|status=]<regex>

As we can see above, filter can be applied on the following fields:

- task id: id
- task name: name
- task category: cat or category
- last result: result
- current status: status

If the filter type is omitted, filter will be applied on all the mentioned fields above.

<regex> must be a valid regular expression. Filtered expression is case insensitive. Foo or foo are equivalent. For further information please visit <http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>

Examples:

- `manage-tasks list -f id=mysql`
- `manage-tasks list -f cat=database`
- `manage-tasks list -f category=database`
- `manage-tasks list -f name=process`
- `manage-tasks list -f process`
- `manage-tasks list -f result=error`
- `manage-tasks list -f status=idle`

It returns 0 as exit status if command completed successfully. Other possible exit status are listed in section [3.2](#).

2.3 disable

The disable command disables a scheduled task.

```
manage-tasks disable <taskId>
```

Where <taskId> is a valid task id. It can be retrieved by launching the list command.

It returns 0 as exit status if command completed successfully or 1 if task is already disabled. Other possible exit status are listed in section [3.2](#).

2.4 enable

The enable command enables a scheduled task.

```
manage-tasks enable <taskId>
```

Where <taskId> is a valid task id. It can be retrieved by launching the list command.

It returns 0 as exit status if command completed successfully or 1 if task is already enabled. Other possible exit status are listed in section [3.2](#).

2.5 info

The info command gives information of a scheduled task.

```
manage-tasks info <taskId>
```

Where <taskId> is a valid task id. It can be retrieved by launching the list command.

Here is an example of the output result:

```
[root@apg-server]# bin/manage-tasks.sh info property-store/Default/import-properties.task
Information on task 'property-store/Default/import-properties.task':
```

Property	Value
Name	import-properties-Default
Current Status	idle
Next Run	Jan 1, 2013 11:00:00 AM
Last Launch	Jan 1, 2013 10:00:00 AM
Last Result	success
Last Successful Completion	Jan 1, 2013 10:00:01 AM
Last Error Encountered	
Last Duration	1s 512ms
Average Duration	1s 56ms
Number of Successful Execution	326
Cron	0 * * * *

It returns 0 as exit status if command completed successfully. Other possible exit status are listed in section [3.2](#).

2.6 start

The start command allows to start a scheduled task.

```
manage-tasks start <taskId>
```

Where <taskId> is a valid task id. It can be retrieved by launching the list command.

It returns 0 as exit status if command completed successfully or 1 if task is already running. Other possible exit status are listed in section [3.2](#).

2.7 stop

The stop command allows to stop a scheduled task.

```
manage-tasks stop <taskId>
```

Where <taskId> is a valid task id. It can be retrieved by launching the list command.

It returns 0 as exit status if command completed succesfully or 1 if task is not running. Other possible exit status are listed in section 3.2.

3 Troubleshooting

3.1 Logs

Some errors might be displayed before or during any operation. For a more complex description, please refer to the last log file in <APG>/Tools/Task-Scheduler/<instance>/logs, usually named client-0-0.log. This file contains a lot more information than the one displayed on screen.

3.2 Exit Status

Value	Cause
10	Invalid configuration for scheduler webservice
11	Invalid command line parameters
12	Error while calling the scheduler webservice
13	Unknown task
14	Unexpected error while processing
15	Connection failure to Scheduler service
16	No task found (resulting from a search like the list command)