



EMC® Watch4net®

Version 6.4u2

Module Manager Guide

P/N: pending

Rev : 40030

EMC²

Copyright ©2009 - 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	v
Purpose	v
Intended Audience	v
Document Organization	vi
Common Abbreviations and Acronyms	vii
Where to get help	ix
1 System Prerequisites	1
1 Operating Systems	1
2 Required Tools	2
2 Manager Overview	3
1 Directory Layout	3
2 Main Configuration and Binary Files	4
3 Manager Usage	7
1 How to Launch it	7
2 Integrated Help	8
4 Module Listing	11
1 Listing Overview	13
2 Available Packaged Modules	13
3 Installed Modules	15
4 Restoration Points	15
5 Module Information	17
1 Command Description	17
2 Information Overview	18

6	Module Installation	21
1	Command Description	21
1.1	Command Help	21
1.2	Command Arguments	22
2	Installation Steps	23
2.1	Command Call	23
2.2	Global Overview	23
2.3	Step by Step	24
3	Web Applications	25
4	Module Dependencies	25
7	Module Update	29
1	Command Description	29
1.1	Command Help	29
1.2	Command Arguments	30
2	Update Steps	30
2.1	Command Call	31
2.2	Global Overview	31
2.3	Step by Step	32
3	Web Applications	34
4	Module Dependencies	34
8	Module Removal	35
1	Command Description	35
1.1	Command Help	35
1.2	Command Arguments	36
2	Removal Steps	36
2.1	Command Call	36
2.2	Global Overview	37
2.3	Step by Step	37
3	Web Applications	39
9	Module Rollback	41
1	Command Description	41
1.1	Command Help	41
1.2	Command Arguments	42
2	Restoration Steps	42

2.1 Command Call	42
2.2 Global Overview	43
2.3 Step by Step	43
10 Module Services	45
1 Services in Installation, Update or Removal	45
2 Command Description	46
2.1 Command Help	46
2.2 Command Arguments	46
3 Common Usage	47
3.1 Service Properties	47
3.2 Service Calls	49
4 Unix Runlevels	50
5 Windows Dependencies	50
11 Silent Processing	53
1 Description	53
2 Available Options	54
3 File Configuration	54
4 Manual Modification	55
12 Uninstallation	57
1 Windows Platforms	57
2 Unix Platforms	58
A Errors and Solutions	59
Unexpected Errors [ERROR CODE -1]	59
1 Help is Displayed [ERROR CODE 1]	60
2 Invalid Configuration [ERROR CODE 2]	60
3 Package not Found [ERROR CODE 3]	60
4 No Instance Found [ERROR CODE 4]	61
5 No Restoration Point Found [ERROR CODE 5]	61
6 Dependency Error [ERROR CODE 6]	61
6.1 Missing Dependency	61
6.2 Dependency Required	62
6.3 Conflicting Dependency	62
7 Not Enough Space Left [ERROR CODE 7]	62

8 Specified Instance not Valid [ERROR CODE 8]	62
9 Specified Instance Already Exists [ERROR CODE 9]	63
10Specified Instance Cannot be Used [ERROR CODE 10]	63
11Operation Aborted [ERROR CODE 11]	63
12Module Operation Failed [ERROR CODE 12]	64
13No Service Found [ERROR CODE 13]	64
14No Service Available [ERROR CODE 14]	64
15Invalid Package Architecture [ERROR CODE 15]	64
16Unix: shell-init error	65

Preface

Purpose

This guide describes the common usage of the module manager, which includes the available commands, the resulting output, possible errors and logging system.

Intended Audience

This guide is meant to be read by any person who will manage the modules of APG: installation, update, removal, service management.

Document Organization

1. System Prerequisites	Details the pre-requisites for the module manager to run as expected.
2. Manager Overview	Gives a general overview of the manager layout and files.
3. Manager Usage	Globally describes how to use the module manager command-line tool.
4. Module Listing	Describes the 'list' command, its arguments and output.
5. Module Information	Describes the 'info' command, its arguments and output.
6. Module Installation	Describes the 'install' command, its arguments, output and behavior.
7. Module Update	Describes the 'update' command, its arguments, output and behavior.
8. Module Removal	Describes the 'remove' command, its arguments, output and behavior.

9. Module Rollback	Describes the 'restore' command, its arguments, output and behavior.
10. Module Services	Describes the 'service' command, its arguments, output and behavior.
11. Silent Processing	Details the <i>silent</i> mode of the module manager as well as its different options.
12. Uninstallation	Details the <i>uninstallation</i> procedure.

Common Abbreviations and Acronyms

<APG>	refers to the APG installation directory
<version>	refers to the version of a given module

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

System Prerequisites

1 Operating Systems

The module manager can be installed and used on either one of the following platforms.

- Linux x86
- Linux x64
- Solaris10 sparc
- Solaris10 x64
- Windows Server 2003/2008 x86
- Windows Server 2003/2008 x64

Most operations don't require administrative privileges (read/write access to the installation directory is generally sufficient). However, for system-wide changes, like service installation, administrative privileges are **required**.

WARNING: If not available, the installation of modules should still proceed successfully, but service may not be

fully installed. For instance, on Unix boxes, `/etc/init.d/` or `/etc/rc.d/` files will not be created.

2 Required Tools

There are no specific requirements, as the installer comes with its own Java version, except on Unix platforms where `unzip` and `tar` are mandatory to bootstrap the installation. However, those are pretty common tools and should already be installed on any box, whatever the distribution.

Note: On Solaris platforms, please make sure the `xpg4` tools and a decent version of `bash` are installed. Minimal installations may not contain them.

Also make sure there is enough space left on the drive the module manager is installed into, as the installation process is likely to take up to twice the size of the installer itself, while the auto installation of modules will take much more than that... though 1G_o of available space should be enough.

Manager Overview

1 Directory Layout

Once the installation is complete, the directory layout will be organized around *APG*'s main categories, and inside each category will the installed modules be listed.

`Collecting` is used to store all collectors, filters, and other parts related to the collecting process. `Backends` holds the data aggregation modules. In `Tools` you can find some various management modules. The installed web applications will be available in `Web-Applications` and deployed in the web servers available in `Web-Servers`.

After a basic installation (no auto-installation of any module), there will not be a lot of categories visible, but everything will be ready to accept new modules.

- `bin` which contains the global *APG* binaries (as well as services on Unix platforms),
- `Doc` which contains all available documentation files, most of them in PDF format,
- `Java` which contains the Java runtime used by all *APG* modules,

- Tools/Module-Repository which contains the available (and packaged) modules the manager can install,
- Tools/Module-Manager which contains the module manager itself.

Note: The location where the files containing packaged modules have to be stored can be configured in the manager's own property file. Its defaults is Tools/Module-Repository.

2 Main Configuration and Binary Files

All configuration files are located in Module-Manager/<version>/conf. There are also some specific properties that can be modified in the file module.properties although the default values fit in most of the cases.

```
#=====#
#           This property file only applies to the Module Manager           #
#=====#

# relative to APG_HOME in the main property file
module.home=Tools/Module-Manager/1.0
module.repository=Tools/Module-Repository
```

The only one you might want to modify (or add if not already present) in this file is the communicator selection property. See section [Silent Processing](#) page 53 for a detailed explanation of its purpose. All binaries are located in Module-Manager/<version>/bin.

During the installation bootstrap process, two files from the module manager are copied inside the main APG/bin directory. The global APG configuration file apg.properties and the manager launcher manage-modules.[sh|cmd].

apg.properties will be used by every script or Java application that needs to access global properties such as the Java location or APG installation directory. Properties specific to a module are located in another file named module.properties.

WARNING: All paths in that file have to be written with forward slashes, even on Windows systems!

The module manager can only be accessed through its command-line interface. In order to do so, simply launch the corresponding executable (APG/bin/manage-modules.sh on Unix systems or APG\bin\manage-modules.cmd on Windows systems) with the adequate parameters. An integrated help is also available for every supported command.

3

Manager Usage

1 How to Launch it

There is only one means to launch the module manager tool, and its is through the script `manage-modules` whose extension depends on the current operating system.

Simply call the script wherever the location. For instance, on Windows systems, if under the APG installation directory, in order to list the installed modules, one could issue the following command.

```
C:\Program Files\APG> bin\manage-modules.cmd list installed
```

Depending on your terminal capabilities, the module manager command line may highlight some parts of the resulting output. All samples from this manual will display such highlights though you might not see them while using the manager. However, there is no means to force one behavior or the other.

2 Integrated Help

Launching the manager without any argument or with the `help` one will result in the display of the integrated help. This would look like the following extract.

```
[root@apg-server]# bin/manage-modules.sh help
Module Manager command-line client, version 1.0 (r17360).
Copyright (C) 2009 Watch4Net.

usage: manage-modules.sh <command> [args] [options]

Type 'manage-modules.sh help <command>' for help on a specific command.

Available commands:
list          : lists packages or installed modules
info          : show detailed information about an installed module
install       : installs a new module
update        : updates an installed module
remove        : removes an installed module
restore       : restores an installed module to its previous state
service       : issues a service request
```

Note that help is also available on every command, with more details on its purpose, arguments and options. For instance, help on the `update` command describes the expected arguments and how to recover them.

```
[root@apg-server]# bin/manage-modules.sh help update
update: Updates the identified module. Data files will be left untouched,
while configuration ones may be updated if needed.
usage: update <module-id> [<module-instance>] [<package-name>] [options]

Use 'list installed' in order to get the list of installed modules that can be
updated. The module identifier and instance will both be highlighted in the
resulting list.

Use 'list available' in order to get the list of packaged modules that can be
used for the update. The package name will be the one highlighted at the end
of each line.

Valid options:
--load="/path/to/answers/file"      : uses the given file to recover answers
--save="/path/to/answers/file"      : uses the given file to save given answers
--use="/path/to/answers/file"       : uses the given file to save and recover answers
```

Finally, those help messages will also appear if the command is not sent with the required arguments, which may happen when using the auto-discovery feature (see each section below for more details of what can be discovered, and thus omitted).

4

Module Listing

This command can be used to know which modules are installed and which ones are available. It can also display restore points if some have been created.

Before going any further, we will define three terms that are going to be extensively use through this document.

- **Module:** A module represent a component of APG. The backends, web servers, collectors (XML Collector, SNMP Collector, etc.), filters (Property Tagging Filter, Cross-Referencing filter, etc) and so on are all modules. They can be installed and managed separatly.
- **Instance:** An instance is a "concrete" representation of a module. Despite the fact that multiple instances of some modules is not something very frequent in APG, multiples instances of collectors or filters can be very useful. Each instance of a particular module will have it's own unique name (across all other instances of this particular module) in order to be able to make some manipulation on that specific instance without applying these manipulations to the other instances. By default, the name of an instance of a module is named "Default".
- **Package:** A package represent a version of a module, frozen with a certain code version (revision). The packages are used by the Module Manager when installing or updating instances of modules and to determine when new versions are available.

Besides, those listing contain the identifiers the other commands require. For instance, it contains the names of the packages needed to install a new module. It also contains the module identifier and installed instance needed to perform an update.

```
[root@apg-server]# bin/manage-modules.sh help list
list: Lists available packages, or installed modules. This list
contains the identifiers required to install/modify them.
usage: list [available|installed|restorable] [options]

Valid options:
--top : only lists top-level modules that are not dependencies of others
--match="identifier" : only lists the modules that match the specified
                      identifier. For instance, '--match=">=frontend-4.0"'
                      will return all frontend modules whose version is
                      greater than 4.0, while '--match="frontend"' will
                      return all frontend modules.
```

The `match` option filters the result based on either a module identifier, either a module version, either both.

In order to keep a specific module, for instance the `sm-collector` module, one would write:

```
[root@apg-server]# bin/manage-modules.sh list --match="sm-collector"
```

In order to keep a specific version, for instance all APG `frontend` modules those whose version is strictly above 5.0, one would write:

```
[root@apg-server]# bin/manage-modules.sh list --match=">frontend-5"
```

In order to keep a specific module and version, for instance all `sm-collectors` whose version is at most 5.0, one would write:

```
[root@apg-server]# bin/manage-modules.sh list --match="<=sm-collector-5.0"
```

1 Listing Overview

When using the `list` command without arguments, both available and installed modules are listed, in alphabetical order. Those two kind of listings are described in details in the corresponding subsections [Available Packaged Modules](#) page 13 and [Installed Modules](#) page 15.

```
[root@apg-server]# bin/manage-modules.sh list
Available Modules:
```

Filename (id-version-arch)	Category	Module	Version	OS-Arch	Status
* backend-5.0-linux-x86	: Backends	APG-Backend	v5.0	linux-x86	
* collector-manager-5.0-linux-x86	: Collecting	Collector-Manager	v5.0	linux-x86	
* frontend-5.0-linux-x86	: Web-Applications	APG-Frontend	v5.0	linux-x86	
* java-6.0u10-linux-x86	: Java	Sun-JRE	v6.0u10	linux-x86	
* module-manager-1.0-linux-x86	: Tools	Module-Manager	v1.0	linux-x86	
* mysql-5.0u67-linux-x86	: Databases	MySQL	v5.0u67	linux-x86	
* mysql-data-5.0u67-linux-x86	: Databases	MySQL-Original-Data	v5.0u67	linux-x86	
* sm-collector-5.1-linux-x86	: Collecting	Smarts-Collector	v5.1	linux-x86	
* tomcat-6.0u18-linux-x86	: Web-Servers	Tomcat	v6.0u18	linux-x86	

```

Installed Modules:
```

Identifier	Instance	Category	Name	Version	Revision	OS-Arch	Status
* backend	Default	: Backends	APG-Backend	v5.0	r13794	linux-x86	
* collector-manager	Default	: Collecting	Collector-Manager	v5.0	r14009	linux-x86	
* java	6.0u10	: Java	Sun-JRE	v6.0u10	r13943	linux-x86	
* module-manager	1.0	: Tools	Module-Manager	v1.0	r14007	linux-x86	
* mysql	Default	: Databases	MySQL	v5.0u67	r13941	linux-x86	
* mysql-data	Default	: Databases	MySQL-Original-Data	v5.0u67	r13941	linux-x86	
! sm-collector	Default	: Collecting	Smarts-Collector	v5.0	r13944	linux-x86	can be

```

updated to v5.1
```

This output can be significantly reduced through the use of the `match` option as described in the [Module Listing](#) section page 12.

2 Available Packaged Modules

When using the `available` argument, the packaged modules available in the Module-Repository directory are read, analyzed and then displayed on the screen, alphabetically ordered.

```
[root@apg-server]# bin/manage-modules.sh list available
Available Modules:
```

Filename (id-version-arch)	Category	Module	Version	OS-Arch	Status
* backend-5.0-linux-x86	: Backends	APG-Backend	v5.0	linux-x86	
* collector-manager-5.0-linux-x86	: Collecting	Collector-Manager	v5.0	linux-x86	
* frontend-5.0-linux-x86	: Web-Applications	APG-Frontend	v5.0	linux-x86	
* java-6.0u10-linux-x86	: Java	Sun-JRE	v6.0u10	linux-x86	
* module-manager-1.0-linux-x86	: Tools	Module-Manager	v1.0	linux-x86	
* mysql-5.0u67-linux-x86	: Databases	MySQL	v5.0u67	linux-x86	
* mysql-data-5.0u67-linux-x86	: Databases	MySQL-Original-Data	v5.0u67	linux-x86	
* sm-collector-5.1-linux-x86	: Collecting	Smarts-Collector	v5.1	linux-x86	
* tomcat-6.0u18-linux-x86	: Web-Servers	Tomcat	v6.0u18	linux-x86	

This gives a quick view of the modules that have been downloaded in order to be either installed, either updated. The package filename is displayed at the beginning of each line, and this string is the exact one to use when referencing a specific packaged module.

Note: As described in the column header, the identifier is the whole string that stands before the version number, inside the module filename. For instance, in backend-5.0-linux-x86, the identifier is backend. In the entry collector-manager-5.0-linux-x86, the identifier is collector-manager.

The status column will count the instances installed for each packaged module listed here, as well as the number of modules that could be upgraded through a specific package. For instance, we could see:

```
Filename (id-version-arch) : Category      Module      [...] Status
-----
```

* sm-collector-5.0-linux-x86	: Collecting	Smarts-Collector		installed (2) - update (1)
------------------------------	--------------	------------------	--	----------------------------

That would mean that the module sm-collector is currently installed twice on this server with version 5.0, and that this package could be used to update another instance of sm-collector which is also installed but with a previous version.

This output can also be significantly reduced through the use of the match option as described in the [Module Listing](#) section page 12.

3 Installed Modules

When using the `installed` argument, the modules installed and registered in the module manager will then be read, analyzed and displayed on the screen, alphabetically ordered.

```
[root@apg-server]# bin/manage-modules.sh list installed
Installed Modules:
```

Identifier	Instance	Category	Name	Version	Revision	OS-Arch	Status
* backend	Default	: Backends	APG-Backend	v5.0	r13794	linux-x86	
* collector-manager	Default	: Collecting	Collector-Manager	v5.0	r14009	linux-x86	
* java	6.0u10	: Java	Sun-JRE	v6.0u10	r13943	linux-x86	
* module-manager	1.0	: Tools	Module-Manager	v1.0	r14007	linux-x86	
* mysql	Default	: Databases	MySQL	v5.0u67	r13941	linux-x86	
* mysql-data	Default	: Databases	MySQL-Original-Data	v5.0u67	r13941	linux-x86	
! sm-collector	Default	: Collecting	Smarts-Collector	v5.0	r13944	linux-x86	can be

updated to v5.1

Please note that the identifier and instance name (see the [Module Installation](#) section for a detailed explanation about the role of the instance name) have been highlighted in the command output. Those identifiers uniquely identify an installed module and thus need to be provided for update or removal operation.

The status column indicates if a newer version of a module is available for an already installed module. If so, the module can be updated.

This output can also be significantly reduced through the use of the `match` option as described in the [Module Listing](#) section page 12.

4 Restoration Points

Restoration points cannot be created manually. They are snapshots of most of a module files at a given time, generated during update or removal operations in order to provide a rollback option.

```
[root@apg-server]# bin/manage-modules.sh list restorable
Available Restore Points:
```

Identifier	Instance	Version	Date
* collector-manager	Default	v5.0	9/09/09 11:43:20 AM

* sm-collector	Default	v5.0	9/09/09	10:41:12	AM
* sm-collector	Default	v5.1	9/09/09	11:15:57	AM

Restoration points can then be used to rollback a module after an update or removal. The required identifiers are the module identifier and instance name, highlighted in the result, as well as the version to rollback to. Please see the section [Module Rollback](#) page 41 for a detailed explanation of the restoration process.

This output can also be significantly reduced through the use of the `match` option as described in the [Module Listing](#) section page 12.

Module Information

This command can be used to display detailed information on an installed module, showing module properties and its files status.

1 Command Description

```
[root@apg-server]# bin/manage-modules.sh info
info: Displays information on an installed module, as well as the installed files status.

usage: status <module-id> [<module-instance>]

Use 'list installed' in order to get the list of installed modules that you can get
information on.
```

This command expects a module identifier to select which package information should be displayed:

- *module-id*, which is the identifier of the installed module that will be displayed (as listed through the “list installed” command [page 15](#)),
- *module-instance*, which is the instance of the installed module that will be displayed (as listed through the “list installed” command [page 15](#)), but is not mandatory if only one instance exists.

2 Information Overview

The following example shows a typical output from the `info` command. In this example, we have requested the information for the `datastore` module:

```
[root@apg-server]# bin/manage-modules.sh info datastore
Module Information:

  Identifier Instance : Category  Module Name  Version Revision OS-Arch  Status
  -----
* datastore  Default   : Databases APG-Datastore v3.2u2  r6013  linux-x64

Files Changes:

  Filename                                File Type      File Status
  -----
* data/BH RSI Repository                  new
* data/bh.err                             Data           modified
* data/bh.out                             new
* data/brighthouse.ini                   Configuration  modified
* data/brighthouse.log                   new
* data/brighthouse.seq                   new
* data/events                             new
* data/mysql/db.MYD                       Data           modified
* data/mysql/db.MYI                       Data           modified
* data/mysql/user.MYD                     Data           modified
* data/mysql/user.MYI                     Data           modified
* data/netflow                            new
* data/test                               Data           deleted
* data/updater.log                        new
```

The first part of the output (“Module Information”) shows module properties. This is very similar to output of the “list installed” command (described on [page 15](#)) but for a single module.

The second part (“Files Changes”) shows every file that have been modified, added or removed since the installation of the package. This is especially useful for Configuration files when you want

to verify whether or not you manually edited them. When a `File Type` is omitted, the file doesn't have any specific type or is unknown for the module manager (typically `new` files).

6

Module Installation

When new modules are made available, they are distributed in a packaged format, usually using the .pkg extension. Once copied in the Module-Repository directory, the manager is able to install them through the adequate command.

1 Command Description

1.1 Command Help

```
[root@apg-server]# bin/manage-modules.sh help install
install: Installs a specific module with the given instance name.
usage: install <package-name> [<instance-name>] [options]

    Use 'list available' in order to get the list of packaged modules that can be
    installed. The package name will be the one highlighted at the end of each
    line.

Valid options:
--load="/path/to/answers/file" : uses the given file to recover answers
--save="/path/to/answers/file" : uses the given file to save given answers
```

```
--use="/path/to/answers/file" : uses the given file to save and recover answers
--standalone                  : prevents automatic dependency installation
```

Answers files will be discussed later, in the [Silent Processing](#) section, page 53.

1.2 Command Arguments

This command takes up to two arguments:

- *package-name*, which is the filename of the package to install, without its extension (as listed through the “list available” command page 13),
- *instance-name*, which is optional, and uniquely identifies the installation for this module. Note that the characters one can use in an instance name are restricted; the name has to match the regular expression “[a-zA-Z0-9][a-zA-Z0-9\._-#]*”, which means every string that begins with a letter or a digit, and then is followed by any number of letters, digits, dots, dashes, underscores, spaces or number signs.

Note: Although the package filename usually contains the module identifier and version in its filename, this information is never directly used by the manager.

The package name does not need to be fully specified. Indeed, the module identifier may be enough to identify the package. The command will look for the latest version of that module in the repository, and if found, will start the installation of this package. However, if two packages for this module have the same (and latest) version, the command will not know which one to select, and thus fail.

For instance, if the repository contains both `collector-manager-5.0-linux-x86` and `collector-manager-4.3u3-linux-x86`, then the command “install collector-manager” will start the installation of `collector-manager-5.0-linux-x86`. However, if the package `collector-manager-5.0-linux-x64` is also present in the repository, then no installation will start.

The instance name may not be specified, in which case, a default one will be used. Note that this can only apply on one installation of this particular module as other installations of the same module

will require a different instance name.

The packaged modules contain all the information required to either install a new module, update one installed or remove it. They contain all binary and configuration files required to run the module, as well as service properties if this very module can be started as a service. No other file is needed.

2 Installation Steps

In this subsection, the installation of the Collector-Manager module will be followed, and any step involved will be described in details, when the `--standalone` option is used.

2.1 Command Call

The most basic way to call such an installation is through the module identifier with the built-in instance name.

```
[root@apg-server]# bin/manage-modules.sh install collector-manager --standalone
```

Calling the manager with the full package name and the 'Default' instance name will provide exactly the same result. This would have been the way to proceed with 5.0 installation if a more recent version (5.1 for instance) would have been present in the module repository.

```
[root@apg-server]# bin/manage-modules.sh install collector-manager-5.0-linux-x86 Default
```

2.2 Global Overview

Applied on a Linux x86 box, this will result in the following steps. Please note that a windows installation would have had almost the same output, except for the highlights, path and service IDs.

```
[root@apg-server]# bin/manage-modules.sh install collector-manager
Starting installation of Collector-Manager v5.0 from collector-manager-5.0-linux-x86...
* Gathering information...
* 'Collector-Manager v5.0' will be registered with instance name 'Default'.
* It will be installed in '/opt/APG/Collecting/Collector-Manager/Default'.
? Continue? [y]es, [n]o > y
* Unpacking files...
* Installing files... 100%
* 43 files have been installed.
* Finalizing installation...
* Installing service 'collector-manager Default'... [ OK ]
Installation successful!
```

2.3 Step by Step

Let's describe each of the involved steps.

```
* Gathering information...
```

First, the manager gathers information about the requested module: it reads the package file and extracts some information from it. This essentially tests the validity of the module file.

```
* 'Collector-Manager v5.0' will be registered with instance name 'Default'.  
* It will be installed in '/opt/APG/Collecting/Collector-Manager/Default'.  
? Continue? [y]es, [n]o >
```

If the file appears valid, then it summarizes the operation that will be processed and asks a final confirmation. In those three lines, we can see the selected package and instance name, as well as the location where the module will be installed.

Please note that the path is created using the module category “Collecting”, name “Collector-Manager”, and instance “Default”. As a consequence, manually finding an installed module is quite easy.

After this step, the manager will check the module dependencies (for instance, a collector installation would need an installed manager), and the remaining space on the selected directory.

```
* Unpacking files...  
* Installing files... 100%  
* 43 files have been installed.
```

Packages files are being unpacked and installed, with the overall progression being displayed at the end of the line. Once finished, the total number of installed files is then provided.

```
* Finalizing installation...  
* Installing service 'collector-manager Default'... [ OK ]
```

Finally, the last steps are processed, and as this module comes with a service file, the corresponding entry is created. It will now be available in both <APG>/bin and /etc/init.d/. In this particular case, it cannot be started yet as there are no collectors installed.

3 Web Applications

Web applications may be deployed on the available web servers when they are installed. This will require a stop of the corresponding service in order to be processed.

```
! WARNING: The service 'apg-tomcat-default' is about to be stopped.
? Continue? [y]es, [n]o > y
* Stopping apg-tomcat-default... [ OK ]
* Deploying web application '/opt/APG/Web-Applications/APG-Frontend/APG/webapps/APG.war'
  on Tomcat-Default...
! WARNING: The web server Tomcat-Default will need to be started again after this process
.
```

If a web application has already been deployed with that name, then the following error message will be displayed:

```
* Web application '/opt/APG/Web-Applications/APG-Frontend/APG/webapps/APG.war' cannot be
  deployed as 'APG.war' on web server Tomcat-Default: there is already an application that
  has been deployed with this name!
```

Indeed, web application deployment uses the instance that has been provided on installation. If a conflict is detected and the deployment is not processed, you can still deploy the application manually. It only requires three simple steps: stop the targeted tomcat server, copy the file in its webapps directory with a name of your choice, and then start the tomcat server again.

4 Module Dependencies

In the previous example, we used the `--standalone` option in order to prevent automatic dependency management and focus on the installation process only. If one of the dependencies would have been missing, the installation would have issued a warning and stop the installation.

```
Starting installation of Transaction-Collector v1.3 from transaction-collector-1.3-linux-
x86...
* Gathering information...
! ERROR: Dependency '>=collector-manager-5.1' is required by Transaction-Collector v1.3!
! ERROR: Dependencies have to be installed before the module.
! WARNING: Please use the module manager to install the requested dependencies.
```

However, the module manager command line is able to handle dependencies, discover the installed ones, and choose to install the missing ones when required.

```

Modules to be processed, in the specified order:
[1] java '6.0u26' v6.0u26
[2] module-manager '1.2' v1.2
[3] U collector-manager 'Default' v5.0 => v5.1
[4] I transaction-collector 'Default' (none) => v1.3
> 2 not modified, 1 to update, 1 to install
> 15.3 MB space required / 21.1 GB available
? Enter the step to modify, 'yes' accept them, or 'no' to cancel the operation [yes] >

```

By entering the number of one step, depending on the step status and available choices, different options may be displayed.

- modify the currently selected installed module (although it has no real impact)
- choose to update an existing module
- change to existing module to update, and choose another one
- change the package used for the installation or update
- choose to install a new instance of a required module
- select another instance name for an installation

In this example, the proposed update of the collector manager will be replaced by the installation of a new instance with a more recent version.

```

The installed instance 'Default' v5.0 of collector-manager will be updated to version 5.1
using 'collector-manager-5.1-linux-x64.pkg'.
Which action do you want to perform?
[4] choose to install another instance
? Please enter your choice, or '0' to go back to previous menu [0] > 4
[1] Collector-Manager 5.1 'collector-manager-5.1-linux-x64.pkg'
? Please select the package to use [1] > 1
? Please enter the instance name with which to install the module [Default] > Transaction

```

Which then results in the following steps.

```

Modules to be processed, in the specified order:
[1] java '6.0u26' v6.0u26
[2] module-manager '1.2' v1.2
[3] I collector-manager 'Transaction' (none) => v5.1
[4] I transaction-collector 'Default' (none) => v1.3
> 2 not modified, 2 to install
> 15.3 MB space required / 21.1 GB available
? Enter the step to modify, 'yes' accept them, or 'no' to cancel the operation [yes] >

```

Finally, each step will be processed one after the other. Modules in green with no action will not be modified. Modules in yellow with either I (install) or U (update) action will be either installed or updated.

Module Update

When new version of existing modules are made available, they are distributed in a packaged format, usually using the `.pkg` extension. Once copied in the `Module-Repository` directory, the manager is able to update the previously installed ones through the `adequate` command.

1 Command Description

1.1 Command Help

```
update: Updates the identified module. Except if specified by the module  
itself, data files will be left untouched, while configuration ones may be  
updated if needed.
```

```
usage: update <module-id> [<module-instance>] [<package-name>] [options]
```

```
Use 'list installed' in order to get the list of installed modules that can be  
updated. The module identifier and instance will both be highlighted in the  
resulting list.
```

```
Use 'list available' in order to get the list of packaged modules that can be  
used for the update. The package name will be the one highlighted at the end
```

```
of each line.  
  
Valid options:  
--load="/path/to/answers/file" : uses the given file to recover answers  
--save="/path/to/answers/file" : uses the given file to save given answers  
--use="/path/to/answers/file" : uses the given file to save and recover answers  
--standalone : prevents automatic dependency installation
```

Answers files will be discussed later, in the [Silent Processing](#) section, page 53.

1.2 Command Arguments

This command takes up to three arguments:

- *module-id*, which is the identifier of the installed module that will be updated (as listed through the “list installed” command page 15),
- *module-instance*, which is the instance of the installed module that will be updated (as listed through the “list installed” command page 15), but is not mandatory if only one instance exists,
- *package-name*, which is the filename of the package to use for the update, without its extension (as listed through the “list available” command page 13).

The package name does not need to be fully specified. Indeed, the module identifier may be enough to identify the package. The command will look for the latest version of that module in the repository, and if found, will start the update to this version. However, if two packages for this modules have the same (and latest) version, the command will not know which one to select, and thus fail.

2 Update Steps

In this subsection, the Smarts-Collector module update will be followed, and any step and process involved will be described in details. We will start with an installed 5.0 version and update it to the 5.1 one.

2.1 Command Call

The most basic way to call such an update is with the module identifier only, assuming there is only one instance installed for that module.

```
[root@apg-server]# bin/manage-modules.sh update sm-collector
```

Calling the manager with the instance name and full package name will provide exactly the same result.

```
[root@apg-server]# bin/manage-modules.sh update sm-collector Default sm-collector-5.1-linux-x86
```

Also, please note that if this collector would have been installed at least twice, with instance names such as Customer A and Customer B, one would have had to call the manager as follows in order to select the Customer B instance. Again, the package name may not be mandatory.

```
[root@apg-server]# bin/manage-modules.sh update sm-collector "Customer B"
```

2.2 Global Overview

Applied on a Linux x86 box, this will result in the following steps. On a Windows one, highlights, paths and service IDs would differ.

```
[root@apg-server]# bin/manage-modules.sh update sm-collector
Starting update of sm-collector Default from v5.0 to v5.1...
* Gathering information...
* Module found in '/opt/APG/Collecting/Smarts-Collector/Default'.
* It will now be updated using 'sm-collector-5.1-linux-x86'.
? Continue? [y]es, [n]o > y
* Unpacking files...
* The file 'conf/smarts-pm.xml' is about to be updated.
? Do you want the update to be automatically performed? [y]es, [n]o, [A]ll, [N]one > A
* Config file 'conf/smarts-pm.xml' has been updated successfully.
? Do you want to keep the original one? [y]es, [n]o, [A]ll, [N]one > N
* Config file 'conf/smarts-acm2.xml' has been updated successfully.
* Updating files... 100%
* 26 files have been updated.
* Finalizing update...
Update successful!
```

2.3 Step by Step

Let's describe each of the involved steps.

```
* Gathering information...
```

First, the manager gathers information about the requested module: it looks for information about the module. This essentially tests the validity of the selected module and package file.

```
* Module found in '/opt/APG/Collecting/Smarts-Collector/Default'.
* It will now be updated using 'sm-collector-5.1-linux-x86'.
? Continue? [y]es, [n]o > y
```

If the module is found and the provided package is a valid one, a final confirmation will be asked. This displays the current location of the installed module as well as the basename of the package file.

After this step, the manager will check the module dependencies and the remaining space left on the module directory. If any issue is encountered, the update will stop.

```
* Unpacking files...
* Updating files...
```

The manager starts unpacking the files, and then updates them. During the update process, all standard files (libraries, binaries, etc...) will be removed or overwritten. However, this behavior does *not* apply for two types of files: the *data* files and the *config* files.

Note: Data files are left untouched. Configuration files will not be overwritten. A backup will be created, and they may be programmatically updated.

```
* The file 'conf/smarts-pm.xml' is about to be updated.
? Do you want the update to be automatically performed? [y]es, [n]o, [A]ll, [N]one > A
* Config file 'conf/smarts-pm.xml' has been updated successfully.
? Do you want to keep the original one? [y]es, [n]o, [A]ll, [N]one > N
```

In this example, two configuration files need to be updated. By answering (yes to) `All` to the first question, one can avoid having to repeat the same choice for all those files. The same applies for the second one, but with `None` this time.

Note: `All` and `None` only apply to one particular set of questions at a time. Other sets will require new answers.

The first question let the manager programmatically issue the update, using whatever means defined in the package itself. The second one was about keeping the backup automatically created before the update. If the configuration file is critical, then this file can be kept as a fallback. The syntax its name is built on is usually:

```
<former-basename>-old-<version>_<date>-<revision>.<former-extension>
```

As a side-note, if an update is required but is not processed, whatever the reason (the user answered `no` at the first question, or it cannot be automatically done, ...), then the following sentence will appear:

```
* The file 'smarts-acm2-mine-5.0.xml' has to be manually migrated to 'conf/smarts-acm2.xml'.
```

You will then have to manually migrate the old configuration file from its backup file `smarts-acm2-mine-5.0.xml` into the new one that kept the real filename.

```
* Updating files... 100%
* 26 files have been updated.
* Finalizing update...
```

When the overall progression reaches 100%, then the number of files that have been updated is displayed and the process ends. In that very case, if the collector manager would have been running during the update, then the manager would have asked whether to stop it or not in order to reload the new definition.

WARNING: On Windows systems, the collector manager needs to be stopped before updating a collector, filter or connector in use. Windows does not support hot replacement of files, and the update is very likely to fail if the collecting component is running.

3 Web Applications

Web applications that have been deployed on the available web servers when they have been installed may be updated too. This will require a stop of the corresponding service in order to be processed.

```
! WARNING: The service 'apg-tomcat-default' is about to be stopped.
? Continue? [y]es, [n]o > y
* Stopping apg-tomcat-default... [ OK ]
* Undeploying web application '/opt/APG/Web-Applications/APG-Frontend/APG/webapp/APG.war'
  from Tomcat-Default...
! WARNING: The web server Tomcat-Default will need to be started again after this process
.

[...]

! WARNING: The service 'apg-tomcat-default' is about to be stopped.
? Continue? [y]es, [n]o > y
* Stopping apg-tomcat-default... [ OK ]
* Deploying web application '/opt/APG/Web-Applications/APG-Frontend/APG/webapps/APG.war'
  on Tomcat-Default...
! WARNING: The web server Tomcat-Default will need to be started again after this process
.
```

If a web application has already been deployed with that name, then the following error message will be displayed:

```
* Web application '/opt/APG/Web-Applications/APG-Frontend/APG/webapps/APG.war' cannot be
  deployed as 'APG.war' on web server Tomcat-Default: there is already an application that
  has been deployed with this name!
```

Indeed, web application deployment uses the instance that has been provided on installation. If a conflict is detected and the deployment is not processed, you can still deploy the application manually. It only requires three simple steps: stop the targeted tomcat server, copy the file in its webapps directory with a name of your choice, and then start the tomcat server again.

4 Module Dependencies

The module update command also supports the automated management of module dependencies. Its behavior is the same as the one described for the installation of a module on page 25. See this section for more details about the display and offered options.

8

Module Removal

The removal of an installed module does not require any additional file. It will delete any installed file, except configuration and data ones which will be preserved. Other files that have been created (manually or programmatically) after the installation will be left untouched. As a consequence, it is very likely that the module installation directory will not be empty and thus remain after this process.

1 Command Description

1.1 Command Help

```
remove: Removes the identified module. Data files will be left untouched,
while the choice will be offered on configuration ones.
usage: remove <module-id> [<module-instance>] [options]

    Use 'list installed' in order to get the list of installed modules that can be
    removed. The module identifier and instance will both be highlighted in the
    resulting list.

Valid options:
  --load="/path/to/answers/file" : uses the given file to recover answers
```

```
--save="/path/to/answers/file" : uses the given file to save given answers  
--use="/path/to/answers/file"  : uses the given file to save and recover answers
```

Answers files will be discussed later, in the [Silent Processing](#) section, page 53.

1.2 Command Arguments

This command takes up to two arguments:

- *module-id*, which is the identifier of the installed module that will be removed (as listed through the “list installed” command page 15),
- *module-instance*, which is the instance of the installed module that will be removed (as listed through the “list installed” command page 15), but is not mandatory if only one instance exists.

If only one instance of that module has been installed, the module-instance parameter does not need to be provided. Indeed, the manager will automatically pick that one.

2 Removal Steps

In this subsection, the Collector-Manager module will be removed, and any step and process involved will be described in details.

2.1 Command Call

The most basic way to call a removal is with the module identifier only, assuming there is only one instance installed for that module.

```
[root@apg-server]# bin/manage-modules.sh remove collector-manager
```

Calling the manager with the instance name will provide the same result. This would also have been the way to proceed if more than one instance of this module would have been installed on the system.

```
[root@apg-server]# bin/manage-modules.sh remove collector-manager Default
```

2.2 Global Overview

Applied on a Linux x86 box, this will result in the following steps. Please note that on a windows one, highlights, paths and service IDs would differ.

```
[root@w4n-dl-apg-qualif APG]# bin/manage-modules.sh remove collector-manager
Starting removal of collector-manager Default...
* Gathering information...
* The module installed in '/opt/APG/Collecting/Collector-Manager/Default' will now be
removed.
? Continue? [y]es, [n]o > y
* Removing apg-collector-manager-default... [ OK ]
* Configuration file 'conf/logging.properties', which has not changed since its
installation, is about to be removed.
? Remove it? [y]es, [n]o, [A]ll, [N]one > A
* Configuration file 'conf/collecting.xml', which has been modified since its
installation, is about to be removed.
? Remove it? [y]es, [n]o, [A]ll, [N]one > n
* Removing files... 100%
* 43 files have been removed.
* 2 files were protected - 3 files were left behind.
* Finalizing removal...
Removal successful!
```

2.3 Step by Step

Let's describe each of the involved steps.

```
* Gathering information...
```

Again, the manager starts with the gathering of information about the module to remove. It essentially tests the properties of the installed modules and the ones that depend on it.

```
* The module installed in '/opt/APG/Collecting/Collector-Manager/Default' will now be
removed.
? Continue? [y]es, [n]o > y
```

When the module has been properly loaded, the manager reminds the user of its current location, and asks for a final confirmation.

```
* Removing apg-collector-manager-default... [ OK ]
```

If the service would have been started, the manager would have asked whether to stop it and continue or not stop it and abort. This would have looked like:

```
! WARNING: The service 'apg-tomcat-default' is about to be stopped.
? Continue? [y]es, [n]o > y
* Stopping apg-collector-manager-default... [ OK ]
* Removing apg-collector-manager-default... [ OK ]
```

```
* Configuration file 'conf/logging.properties', which has not changed since its
installation, is about to be removed.
? Remove it? [y]es, [n]o, [A]ll, [N]one > A
* Configuration file 'conf/collecting.xml', which has been modified since its
installation, is about to be removed.
? Remove it? [y]es, [n]o, [A]ll, [N]one > n
* Removing files... 100%
```

During the file removal process, two types of questions can be asked about configuration files.

The first one in the example above applies on configuration files that have not been modified since their installation. It is not really necessary to keep them as they can be recovered through a brand new installation of the same module. This is why it was answered *Yes to All* at the corresponding removal question.

However, the second type of question deals with modified configuration files, which ones should be kept if this module is planned to be reinstalled again. It was decided to keep at least `collecting.xml` in this example, and as it was the only modified file, no other question was asked.

```
* 43 files have been removed.
* 2 files were protected - 3 files were left behind.
* Finalizing removal...
```

Then follows the summary of removed files, as well as protected and untouched ones.

2 files were protected means two data or configuration files were not removed. As only one configuration file was protected (`collecting.xml`), then the remaining one must be a data file.

3 files were left behind means three additional files were created by the application. In this case, it is very likely that they are logging files located in the `log` directory.

As a consequence of these files being kept on the filesystem, the module installation directory will not be empty at the end of the removal, and thus remain.

Note: If no file were manually protected by answering a question and if, however, some of them were kept,

then you can safely remove the module installation directory mentioned at the beginning of the removal operation.

3 Web Applications

Web applications that have been deployed on the available web servers when they have been installed may be removed too. This will require a stop of the corresponding service in order to be processed.

```
! WARNING: The service 'apg-tomcat-default' is about to be stopped.
? Continue? [y]es, [n]o > y
* Stopping apg-tomcat-default... [ OK ]
* Undeploying web application '/opt/APG/Web-Applications/APG-Frontend/APG/webapp/APG.war'
  from Tomcat-Default...
! WARNING: The web server Tomcat-Default will need to be started again after this process
.
```


Module Rollback

The creation of a restoration point is an automated process only. It cannot be done manually. In fact, it is triggered by two events: the update and the removal of a specific module.

The goal of such saves it to be able to rollback a module in a previous state if the update does not prove successful, or if the removal was a mistake. However, in some very particular cases, it might not be possible. Indeed, a restoration point does not keep the data files as they are either not relevant or too big to be kept. If those ones are updated or manually removed, the changes may be irreversible as a module cannot be downgraded. Data backup is left to the customer's discretion.

1 Command Description

1.1 Command Help

```
restore: Restores the identified module to the specified state. Data files  
will *not* be restored, so it is recommended to use a backup for them if they  
don't fit.
```

```
usage: restore <module-id> [<module-instance>] [<version>]
```

```
Use 'list restore' in order to get the list of installed modules that can be
```

```
restored.  
  
Valid options:  
--load="/path/to/answers/file" : uses the given file to recover answers  
--save="/path/to/answers/file" : uses the given file to save given answers  
--use="/path/to/answers/file" : uses the given file to save and recover answers
```

Answers files will be discussed later, in the [Silent Processing](#) section, page 53.

1.2 Command Arguments

This command takes up to three arguments:

- *module-id*, which is the identifier of the installed module that will be restored (as listed through the “list restorable” command page 15),
- *module-instance*, which is the instance of the installed module that will be restored (as listed through the “list restorable” command page 15), but is not mandatory if only one instance exists,
- *version*, which is the version to which the module should be rolled back.

If only one instance of a module has been saved, then the module-instance parameter can be ignored. Besides, if, given one module and instance, only one version exists for them, the version parameter can be ignored too. However, as soon as there are more than one, they have to be specified as arguments of the command.

2 Restoration Steps

In this subsection, the Smarts-Collector, after having been installed in 5.0 version, then updated to 5.1 and finally removed, will be rolled back to its first version: 5.0.

2.1 Command Call

The most basic way to call a restoration is with the module identifier only, assuming there is only one instance and version stored for that module.

```
[root@apg-server]# bin/manage-modules.sh restore sm-collector
```

Calling the manager with the instance name and version will provide the same result. This would also have been the way to proceed if more than one instance or version of this module would have been stored on the system.

```
[root@apg-server]# bin/manage-modules.sh restore sm-collector Default 5.0
```

We'll use the latter one, without the instance name, in this example as we have two restoration points for the Smarts Collector: one for 5.0 and one for 5.1.

2.2 Global Overview

Applied on a Linux x86 box, this will result in the following steps. On a windows one, highlights, paths and service IDs would differ.

```
[root@apg-server]# bin/manage-modules.sh restore sm-collector 5.0
Starting restoration of sm-collector Default v5.0...
* Gathering information...
* Issuing a rollback will overwrite all existing files - except the data ones - and
replace them with their previous versions, including configuration files.
* The module 'Smarts-Collector' will now be restored in '/home/raubert/APG/Collecting/
Smarts-Collector/Default'.
! WARNING: The targeted directory already exists! This may overwrite existing files in
this directory.
? Continue? [y]es, [n]o > y
* Restoring files...
Restoration successful!
```

2.3 Step by Step

Let's describe each of the involved steps.

```
* Gathering information...
```

This step involves the recovery of stored information as well as a quick check on their validity.

```
* Issuing a rollback will overwrite all existing files - except the data ones - and
replace them with their previous versions, including configuration files.
* The module 'Smarts-Collector' will now be restored in '/home/raubert/APG/Collecting/
Smarts-Collector/Default'.
! WARNING: The targeted directory already exists! This may overwrite existing files in
this directory.
```

```
? Continue? [y]es, [n]o > y
```

The usual warning may state (it is the case here) that the targeted directory exists. For instance, this directory may contain the remains of a removed module, or the current module in another version. There is nothing to worry about if it is the right targeted directory.

```
* Restoring files...
```

There is no protection of configuration files during this phase, so no question will be asked until all files have been restored.

10

Module Services

The module manager is also able to manage services for any installed and registered module. This covers the installation of those services as well as their usage. Besides, it provides the means to quickly start or stop all of them in the required order.

1 Services in Installation, Update or Removal

Services will be installed, updated or removed during the module operations. They are usually stopped and removed before any operation on the module, and then installed again when the operation completes.

In some very unlikely cases, it may happen that the service installation or update fails. However, *it does not means the module operation has failed!* Services can be installed and updated separately as they are managed by an independant process. In order to troubleshoot the issue, please refer to the manager log files. Once the issue has been fixed, the command described below can be used to re-install the corresponding service.

2 Command Description

2.1 Command Help

```
service: Issues a specific service request. This could be either to
refresh a service definition, either to use that service. Please note that
services can also be used directly.
usage: service request (<module-id>|AUTO|ALL) [(<module-instance>|ALL)]

    Use 'list installed' in order to get the list of installed modules that can be
    used with this command.

Valid requests:
- install : installs or updates a module service
- update  : updates a module service
- remove  : removes an installed service
- start   : starts an installed service
- stop    : stops an installed service
- restart : restarts an installed service
- status  : gives the status of an installed service

Valid options:
--load="/path/to/answers/file" : uses the given file to recover answers
--save="/path/to/answers/file" : uses the given file to save given answers
--use="/path/to/answers/file"  : uses the given file to save and recover answers
```

Answers files will be discussed later, in the [Silent Processing](#) section, page 53.

2.2 Command Arguments

This command takes up to three arguments:

- *request*, which is the actual service request. Available commands are described in the service help,
- *module-id*, which is the identifier of the installed module on which the service request will apply (as listed through the “list installed” command page 15),
- *module-instance*, which is the instance of the installed module on which the service request will apply (as listed through the “list installed” command page 15), but is not mandatory if only one instance exists.

Note: The `module-id` arguments can be replaced by the keyword `all` or `auto` in lower or upper case. With those, the argument `module-instance` is ignored, and all services (or services configured in automatic startup mode when using `auto`) will receive the specified request. Note that the `module-instance` can also be replaced by the `all` keyword in order to target all instances sharing the same module ID.

3 Common Usage

3.1 Service Properties

The definition of the service that comes with a module stands inside a particular property file, usually available in the `conf` directory of the corresponding module. The manager uses that file for two purposes: first, to know whether there is a service or not for this module, and then to install or update the corresponding module.

Here comes a service property file sample. Note that some properties, such as the one called `$MODULE_HOME$`, will be automatically replaced when the service is installed. The names of those properties are taken from both the configuration files `apg.properties` and `module.properties`, and then are set to upper case, and finally slightly modified so that each `."` is replaced by `_"`. Ex: with `module.home` being defined in a property file, one can then use `$MODULE_HOME$` in the corresponding service file.

```
#=====#
#           This property file only applies to the Tomcat Unix Service           #
#=====#

#=====
# Main Service Properties
#=====
name=APG Tomcat
description=This service manages APG's web-applications
startup=auto
start.order=96
stop.order=4

#=====
# Service Dependencies, starting from index 1
```

```

#=====
#depends.1=

#=====
# Application Classpath, starting from index 1
#=====
classpath.1="$MODULE_HOME$/bin/bootstrap.jar

#=====
# Initial Java Heap Size (in MBytes)
#=====
memory.init=256

#=====
# Maximum Java Heap Size (in MBytes)
#=====
memory.max=512

#=====
# Additional Java Parameters, starting from index 1
#=====
jvm.param.1=-Djava.awt.headless=true
jvm.param.2=-XX:MaxPermSize=128m
jvm.param.3=-Dcatalina.base="$MODULE_HOME$"
jvm.param.4=-Dcatalina.home="$MODULE_HOME$"
jvm.param.5=-Djava.endorsed.dirs="$MODULE_HOME$/endorsed
jvm.param.6=-Djava.io.tmpdir="$MODULE_HOME$/temp
jvm.param.7=-Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.Jdk14Logger
jvm.param.8=-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
jvm.param.9=-Djava.util.logging.config.file="$MODULE_HOME$/conf/logging.properties

#=====
# The service start and stop commands;
# they will be run inside the installation directory
# mode is one of 'java' or 'exec' - and also 'kill' for the stop mode
#=====
start.mode=java
start.target=org.apache.catalina.startup.Bootstrap
start.param.1=main
start.param.2=start
stop.mode=java
stop.target=org.apache.catalina.startup.Bootstrap
stop.param.1=main
stop.param.2=stop
# timeout is in seconds
stop.timeout=10

```

```
#####  
# Default Output  
#####  
output.out=auto  
output.err=auto
```

Note that although this service file is dedicated to Unix, there are not a lot of differences with the windows one. Indeed, most of the attributes are common between the two types of services, except some particular system-dependant settings or calls..

Note: There is usually no need to modify the property files. All module properties should be changed in `module.property`. They will be made available to the module.

WARNING: If this **property file is modified**, then the corresponding **service has to be updated** for the changes to be repercutated on the installed service.

3.2 Service Calls

Services can be called either through the operating system own service management tool, either through the module manager itself. However, the module manager is the only one which is able to install, update or remove a service based on the property file described in the previous section.

Among all the available service requests, the `status` one proves quite usefull when checking which services are installed and which ones are running.

```
* apg-mysql-default           [ running ]  
* apg-backend-default        [ running ]  
* apg-tomcat-default          [ running ]  
* apg-collector-manager-default [ stopped ]
```

4 Unix Runlevels

On unix systems, the runlevels on which the services start or stop can be configured through two properties. They can be specified in either the `JAVA_OPTS` value or the `module.properties` file. Runlevel entries have to be separated with a `','` character.

In `JAVA_OPTS`, it would look like:

```
JAVA_OPTS="${JAVA_OPTS} -Dapg.service.rc.start=3;4;5 -Dapg.service.rc.stop=0;1;2;6"
```

In the `module.properties` file, it would look like:

```
apg.service.rc.start=3;4;5  
apg.service.rc.stop=0;1;2;6
```

Please note that the values listed here are the default ones.

The `init.d` and `rc.d` paths can also be modified through the following properties.

- `apg.service.init.path` which defaults to `/etc/init.d/`
- `apg.service.rc.path` which defaults to `/etc/rc`
- `apg.service.rc.suffix` which defaults to `.d/`

5 Windows Dependencies

On Windows systems, there is no service startup order. As a consequence, if no dependency is created between services, they will start in a random order.

Such a behavior could prevent some APG services from starting. This is why a weak-dependency system has been implemented in the module services.

Let's take the example of the MySQL and Backend modules. The backend will not start if MySQL is not already running. In order to achieve this, the backend service properties contains the `mysql-dependency` line.

```
depends.i=mysql
```

Another possibility could be that two different instances of the mysql database are installed on the same server. One used by the backend (with instance name `Default`), the other one used by another service (with instance name `Test`). Setting the property as explained above would create a

dependency with both mysql instances. So in order to restrict the backend service dependency, one could specify the module instance name as well.

```
depends.1=mysql_Default
```

Note: Check the file located in <module-install-dir>/conf and named `win-service.properties` in order to get those dependencies.

This property is used in order to generate a direct dependency when the backend service is installed *if, and only if*, a mysql service has *already been installed*. The module installation does not really matter as long as the backend service is updated *after* the installation of the MySQL service.

However, in our example, if the MySQL service needs to be removed for any reason, please note that the backend service will need to be updated for the dependency to be removed. Failing to do so will result in the backend service refusing to start.

Note: Make sure to update all remaining services after removing a service that might be a dependency.

Silent Processing

1 Description

The silent processing feature allows default, or replication of, installations, updates or removal without asking any question.

There are two types of silent processing.

- The most basic one uses default answers that have been pre-defined for each question. It is not possible to modify such answers, and they follow what is expected to be the *default* operation.
- The other one uses property files that contain answers for each question that might be asked. Every question is uniquely identified, and needs to be defined in that file.

Note: The generation of such a file should not be done manually, but rather automatically through the `save`

[in answers file] option.

2 Available Options

The default `Communicator` can use an answer file, if given, and asks questions when it does not find answers inside the specified file. Although it will fit most of the cases, another communicator can be selected inside the `module.properties` file from the module manager.

If the corresponding property does not exist yet, simply add as one single line in order to get the default one. This is the one to select for a regular usage of the command line tool.

```
apg.manager.communicator=com.watch4net.apg.module.controller.cli.communicator.FileCliCommunicator
```

In order to get the communicator that always answers the default choice, the following line should be specified. This is the one to select for a silent default processing.

```
apg.manager.communicator=com.watch4net.apg.module.controller.cli.communicator.DefaultCliCommunicator
```

Finally, the last one uses a combination of both the file that contains answers and the default choices of questions. This is the one to use for a silent processing that needs to be driven on some specific questions.

```
apg.manager.communicator=com.watch4net.apg.module.controller.cli.communicator.FileDefaultCliCommunicator
```

3 File Configuration

The answers file can be set to either recover answers, either save new answers, either both. The selection of such a behavior can be achieved through three distinct means. The one described in each command consists in giving the file on the command-line, after every command and arguments.

For instance, one could issue the following command in order to use answers from the file `/opt/APG/answers.properties`

```
[root@apg-server]# bin/manage-modules.sh update sm-collector --load=/opt/APG/answers.properties
```

Those settings can also be configured as system-wide properties, for instance in the `JAVA_OPTS` environment variable, or as module-specific properties in the corresponding manager `module.properties` file.

In the two last cases, the property keys are:

- `apg.manager.communicator.answers.save` for the save file
- `apg.manager.communicator.answers.load` for the load file
- `apg.manager.communicator.answers.use` for the use file

Note: The `use` answers file cannot be used in conjunction with either the `save` or `load` one.

4 Manual Modification

There is no need to manually set or change values in such a save/load/use file. The module manager should be used instead in order to create and modify them as it knows which keys to use and how to set and escape values.

WARNING: Manual modification of the save/load/use files content is not recommended at all.

Changing a value to a non-valid one will automatically discard it, thus asking for an answer as if it were not in the file. The contents of those files is not expected to be easily readable. However, if you still want to have a look at them, please keep in mind that values depend on the questions that have been asked, and as a consequence you may only see integers.

WARNING: Setting unescaped or wrongfully escaped string values might invalidate the whole file: the module manager will not be able to load it anymore!

The best practice is to always run the procedure once in a test environment, so that the answers are saved in such a file, and then use that file on other environments, without modifying it.

Uninstallation

1 Windows Platforms

The windows installer comes with a built-in uninstall process. This operation will remove all installed services and completely wipe out *APG's* installation directory. It can be launched from the following locations.

- either the start menu Programs > APG > Uninstall APG,
- either the Add or Remove Programs tools available in the Control Panel.

2 Unix Platforms

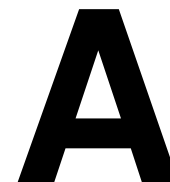
The unix installer does not provide an `uninstall` tool, but the global blind removal is quite easy and can be achieved through the following steps:

1. remove all installed services:

```
<APG>/bin/manage-modules.sh service remove all
```

2. wipe out *APG*'s installation directory:

```
rm -Rf <APG>
```



Errors and Solutions

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/Module-Manager/<version>/logs, usually named `manager-0-0.log`. This file contains a lot more information than the one displayed on screen.

Unexpected Errors [ERROR CODE -1]

Those errors are very unlikely to be displayed, but it may happen that one specific error does not have its own error code, and as a consequence will be labelled as *unexpected...* or not labelled at all

Unexpected errors should contain a self-describing error message, and a detailed trace in the manager log files. However, it might not always be the case. For instance, one that looks like :

```
! ERROR: java.lang.ExceptionInInitializerError
```

is quite obscure. However, the corresponding message in the logs is much more detailed and explains exactly what the issue is.

Fortunately, most of the errors you may encountered have a dedicated error message and code, which are listed in the following sections.

1 Help is Displayed [ERROR CODE 1]

If the command help is displayed when launching the process, then it may be either a syntax error, either the use of the simple form with only a subset of the required arguments linked to fact that the manager was not able to guess the remaining ones.

Please correct the syntax error or give the complete list of arguments.

2 Invalid Configuration [ERROR CODE 2]

The error should look like the following extract.

```
! ERROR: Cannot setup a 'LOAD' or 'SAVE' answers file when a 'USE' file has been
specified!
! WARNING: Please check the 'JAVA_OPTS' environment variable, the 'module.properties'
file and the command-line arguments.
```

A conflict has been detected in the specified answers files. If a 'USE' file has been given, then no other file may be specified. The module manager is not able to guess which one should be used.

Please remove either the 'USE' file or the 'LOAD' and/or 'SAVE' file from the command-line and/or the property files.

3 Package not Found [ERROR CODE 3]

The error should look like the following extract.

```
! ERROR: Package 'collector-manager' not found!
! WARNING: Use the command 'list available' to get a list of all available packages.
```

It means that either there is no package for the collector-manager module. There may be a typo somewhere (it is the case in this example), or the package may simply not exist.

This error message will also appear when using a module identifier instead of the complete file-name when two packages exist for the same module identifier with the exact same version and revision, but not the same targeted operating system or architecture. For instance, after downloading the

collector-manager package for both linux x86 and linux x64, and then calling “install collector-manager”, such an error will be displayed.

4 No Instance Found [ERROR CODE 4]

The error should look like the following extract.

```
! ERROR: No instance name 'DoesNotExist' found for module 'sm-collector'!  
! WARNING: Check that this instance exists, and is correctly spelled.
```

There may be a typo in the specified instance name. Use the command “list installed --match=sm-collector” in order to get the list of installed instances (replace sm-collector with the ID of the module you are targeting). If the instance contains spaces, it needs to be inside double quotes, such as in “update sm-collector “Instance Name With Spaces””.

5 No Restoration Point Found [ERROR CODE 5]

The error should look like the following extract.

```
! ERROR: No restoration point found for module 'sm-collector' instance 'DoesNotExist' in  
v5.0!  
! WARNING: Check that this restoration point exists, and that all attributes are  
correctly spelled.
```

There may be a typo in the specified instance name. Use the command “list restorable --match=sm-collector” in order to get the list of installed instances (replace sm-collector with the ID of the module you are removing). If the instance contains spaces, it needs to be inside double quotes, such as in “restore sm-collector “Instance Name With Spaces””.

6 Dependency Error [ERROR CODE 6]

6.1 Missing Dependency

The error should look like the following extract.

```
! ERROR: Dependency '>=collector-manager-5.0' is required by Smarts-Collector v5.0!  
! ERROR: Dependencies have to be installed before the module.  
! WARNING: Please use the module manager to install the requested dependencies.
```

When one chooses to use the --standalone option when installing or updating a module, this message is to be expected each time a dependency needs to be installed first. In this case, the smarts collector installation requires the collector manager module to be installed in at least the 5.0 version.

6.2 Dependency Required

The error should look like the following extract.

```
! ERROR: Dependency '>=collector-manager-5.0' is required by Smarts-Collector v5.0!  
! ERROR: This module is a required dependency. It cannot be removed.  
! WARNING: Please remove the dependencies first, and then this module.
```

There is currently no auto-removal of modules that depend on another one in the module manager. So this message is to be expected each time an installed module depends on one that is being removed. In this case, the smarts collector needs to be removed before the collector manager.

6.3 Conflicting Dependency

The error should look like the following extract.

```
! ERROR: Dependency '>=collector-manager-5.0' 'Default' is already installed!
```

When installing a module that has a dependency on a specific instance of a specific module, you must ensure that that instance of that module does not already exist, otherwise you will receive this message. To fix the problem, you can either remove the installed module, or you can change the instance name that you wish to install the module with.

7 Not Enough Space Left [ERROR CODE 7]

The error should look like the following extract.

```
! ERROR: There's not enough space left in '/tmp/APG-Module-Manager'!  
! ERROR: 172,271 MB are required while only 35,356 MB are available.  
! WARNING: Please remove unused files or select another location.
```

This may only happen when very little space remains on either the *temporary directory*, or the *working directory*, or the *APG installation directory*. The first one defaults to the OS default temporary directory but can be modified through the system property `java.io.tmpdir`.

8 Specified Instance not Valid [ERROR CODE 8]

The error should look like the following extract.

```
! ERROR: The selected instance name '(Default)' is not valid!  
! WARNING: Please use either the default one either one that matches '[a-zA-Z0-9][a-zA-Z0-9\\.\-_ #]*'.
```

It means the instance does not match the mandatory pattern. This regular expression means every string that begins with a letter or a digit, and then is followed by any number of letters, digits, dots, dashed, underscores, spaces or pound signs. In that case, the instance name (Default) begins with a parenthesis, which is not allowed.

9 Specified Instance Already Exists [ERROR CODE 9]

The error should look like the following extract.

```
! ERROR: Specified instance 'Default' already exists for Collector-Manager!  
! ERROR: Modules can be installed more than once, but with different instance names only.  
! WARNING: Please update the existing instance or select another instance name.
```

It means the collector-manager module is already installed with the instance name Default. If you needed to update this module, please use the `update` command described in the [Module Update](#) section page 29. Else, you may install another instance, but with a different name.

10 Specified Instance Cannot be Used [ERROR CODE 10]

The error should look like the following extract.

```
! ERROR: Specified instance 'Default' cannot be used for Collector-Manager!  
! ERROR: The directory '/opt/APG/Collecting/Collector-Manager/Default' already exists and will not be overwritten.  
! WARNING: Please select another instance name for this module or manually remove this directory.
```

It means the installation path that would have been built using the module category, name and instance already exists. As a measure of precaution, the manager will *never* overwrite an existing directory. If the targeted directory is the right one, but already exists for some reason, then you need to manually remove it, and then relaunch the installation. Else, please provide a different instance name for this module.

11 Operation Aborted [ERROR CODE 11]

The error should look like the following extract.

```
! ERROR: The operation on Collector-Manager v5.0 has been aborted!
```

Such an event may occur for many reasons. The most evident one being a negative answer to a mandatory question such as “Continue?”. It could also be a internal mechanism that prevented the operation from being processed. More information may be found in the corresponding manager log file.

12 Module Operation Failed [ERROR CODE 12]

Such errors are very unlikely to occur and will not exactly display the name specified in this section's title.

```
* Gathering information...  
! ERROR: Unable to access data: File '...' cannot be read!
```

They usually contain enough information to solve the corresponding issue, but as usual, the detailed message will be in the log file.

13 No Service Found [ERROR CODE 13]

The error should look like the following extracts.

```
! ERROR: No service found for 'sm-collector' 'Default'!
```

This means the service has not been installed for this module, either because there is no service defined for it, either because its service has been removed.

14 No Service Available [ERROR CODE 14]

```
! ERROR: No service available for 'sm-collector' 'Default'!
```

This means there is no service available for this module. It usually happens for modules that are only components of a larger modules, and as such cannot be launched separately.

15 Invalid Package Architecture [ERROR CODE 15]

```
! ERROR: Package architecture 'windows-x64' does not match the expected one 'linux-x64'.
```

This means there has been a mismatch between system architecture and the selected package. Please use appropriate package for your system operating system and architecture

16 Unix: shell-init error

```
[root@w4n-dl-apg-qualif conf]# ../../../../bin/manage-modules.sh help
shell-init: error retrieving current directory: getcwd: cannot access parent directories:
No such file or directory
shell-init: error retrieving current directory: getcwd: cannot access parent directories:
No such file or directory
pushd: error retrieving current directory: getcwd: cannot access parent directories: No
such file or directory
```

It can only happen on Unix systems, and means the directory in which you are located does not exist anymore. This may happen if you are inside a module directory, for instance its `conf` directory, and then completely remove that module. In that case, the `conf` directory gets removed but you remain inside.

Such a behavior cannot happen on Windows systems because the OS itself will prevent the directory from being removed if someone has a shell opened in that directory.