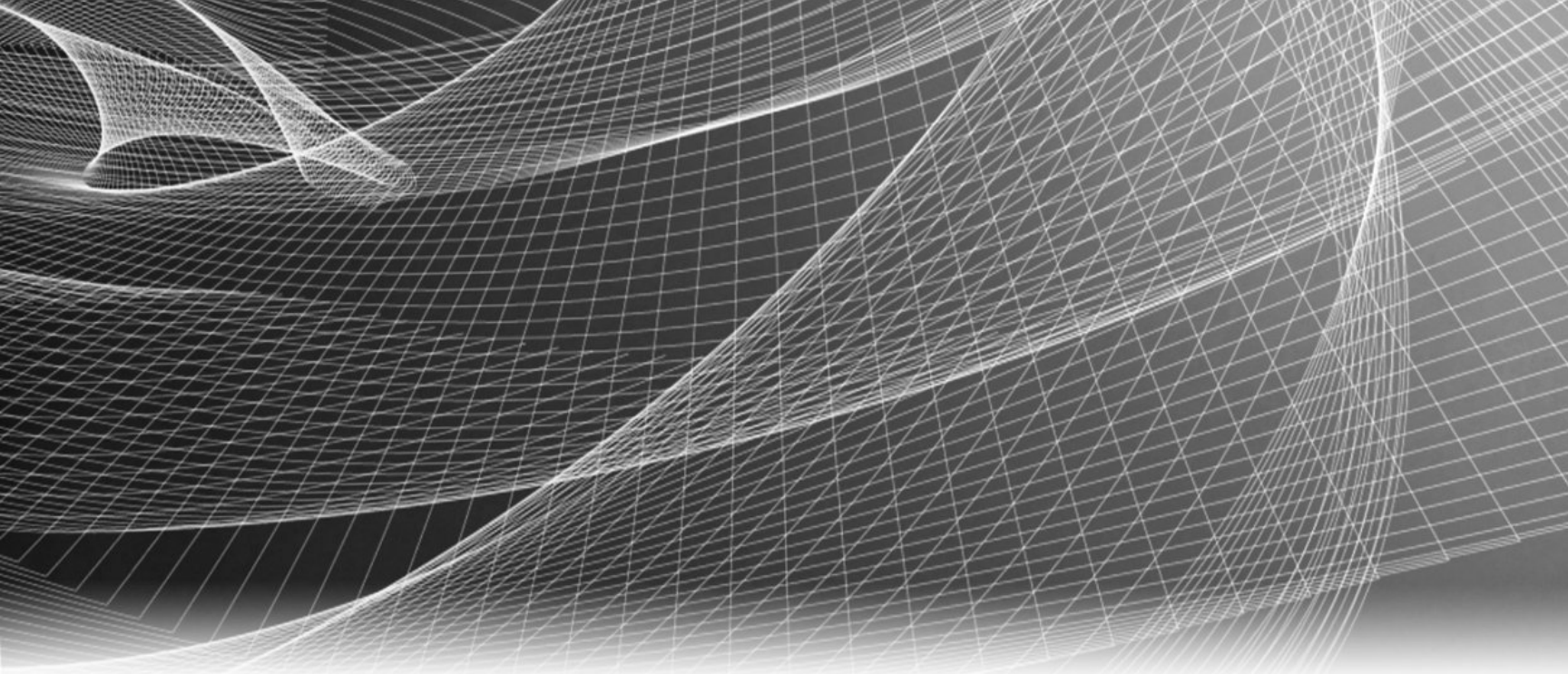




EMC[®] Watch4net[®]

Version 6.3u2

Diagnostic Tools Guide

P/N: pending

Rev : 42621

Copyright ©2013 - 2014 EMC Corporation. All rights reserved. Published in the USA.

Published April, 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
Intended Audience	iii
Document Organization	iii
Where to get help	v
1 Diagnostic	1
2 JMX Fiddler	3
1 Configuration	3
2 Available commands	4
2.1 help	4
2.2 example	4
2.3 list	5
2.4 view	5
2.5 get	5
2.6 call	6

Preface

The purpose of this guide is to provide information about the *Diagnostic Tools* and the way to use it.

Intended Audience

This guide is intended to be read by any administrator or support team who will need to get debug data from *APG* and from Java Virtual Machines.

Document Organization

1. Diagnostic	Using the diagnostic extraction tool.
2. JMX Fiddler	Using the JMX Fiddler.

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

Diagnostic

The diagnostic tool is used to extract debug data from the system and all configuration and log files from modules in the *APG* install. All extracted data is compressed in a file saved in `/tmp` or in *APG* directory depending on the operating system.

The script is located in `<APG>/Tools/APG-Diagnostic-Tools/<instance>/bin`. It is named `diagnostic.sh` for linux and solaris systems, and `diagnostic.cmd` for windows ones.

2

JMX Fiddler

The JMX-Fiddler is used to get data and call methods on JVMs (Java Virtual Machines). It can fetch debug data from all Java based *APG* modules.

The script is located in `<APG>/Tools/APG-Diagnostic-Tools/<instance>/bin`. It is named `jmx.f.sh` for linux and solaris systems, and `jmx.f.cmd` for windows ones.

1 Configuration

If your terminal and its font support UTF-8, it is possible to get a nicer result tree structure by enabling `jmx.fiddler.ascii.mode=false` in `module.properties` file in the `conf` directory.

WARNING: For Windows systems, the `jmx.f.cmd` script uses PsExec (<http://technet.microsoft.com/en-ca/sysinternals/bb897553.aspx>) to run as system user. Make sure PsExec is in the Path to make it work. PsExec will show the error code of the command after executing it. 0 is the no error code.

2 Available commands

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

Usage for windows: `jmx-f.cmd <command> [<command args> ...]`

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

2.1 help

The help command gives usage for each command.

```
jmx-f help [<command>]
```

Where `<command>` is a command name.

2.2 example

The example command gives some examples to use the JMX-Fiddler.

```
jmx-f example
```

Here are the provided examples:

View some JVM data. Including classpath, system properties, inputs and uptime.

```
jmx-f view <vm id> -t -o "java.lang:type=Runtime"
```

View memory data about heap and non heap.

```
jmx-f view <vm id> -t -o "java.lang:type=Memory"
```

View some operating system data.

```
jmx-f view <vm id> -t -o "java.lang:type=OperatingSystem"
```

Dump heap in file.

```
jmx-f call <vm id> -c dumpHeap -m <output file> -m true -o "com.sun.management:type=HotSpotDiagnostic"
```

Dump all threads here.

```
jmxof call <vm id> -c dumpAllThreads -m true -m true -o "java.lang:type=Threading"
```

Call garbage collection. JVM may consider doing it.

```
jmxof call <vm id> -c gc -o "java.lang:type=Memory"
```

2.3 list

The list command gives a list of all available JVMs to get data from.

```
jmxof list [-v] [-v]
```

By default, the command gives only minimal information about the JVM (pid and maybe class name). Adding one `-v` will add the directory where the process was started. Adding two will also give the classpath.

2.4 view

The view command shows exposed MBean attributes and operations. Displayed as a tree structure.

```
jmxof view <vm id> [-a] [-o <object name>] [-p] [-t]
```

- `<vm id>` : The JVM id. (Can be obtained from the list command)
- `-a` : Shows attributes and operations. Same as `-p` and `-t` combined.
- `-o <object name>` : The requested `ObjectName`. Filters the results to only the provided `ObjectName`.
- `-p` : Shows operations.
- `-t` : Shows attributes.

2.5 get

The get command shows attribute value.

```
jmxof get <vm id> -g <attribute name> [-o <object name>]
```

- `<vm id>` : The JVM id. (Can be obtained from the list command)
- `-g <attribute name>` : The attribute name to show. If no `ObjectName` is specified, it will show all matching attributes.
- `-o <object name>` : The requested `ObjectName`. Filters the results to only the provided `ObjectName`.

2.6 call

The call command calls an operation.

```
jmxr call <vm id> -c <operation name> [-m [<type>:]<param value> ...] [-o <object name>]
```

- `<vm id>` : The JVM id. (Can be obtained from the list command)
- `-c <operation name>` : The operation name to call. If no `ObjectName` is specified, it will try to find a single match. If more than one match is found, the list of matching operation will be shown.
- `-m [<type>:]<param value>` : A param for operation call. Type is only required when an other operation has the same name and the same number of parameters. Type is shown when using `view` command.
- `-o <object name>` : The requested `ObjectName`. Filters the results to only the provided `ObjectName`.

WARNING:

If the parameter value contains a colon (:), the type must be specified.

Note:

Array parameter : An array is written between square brackets "[" and "]", and values are comma separated ",". For example, an array of long with its type declared: `-m [J: [1,2,3]`.

Here is the meaning of base types:

Base type character	Type	Details
B	byte	
C	char	
D	double	double-precision floating-point value
F	float	single-precision floating-point value
I	int	
J	long	
L classname ;	reference	Example: Ljava.lang.String;
S	short	
Z	boolean	true or false
[	reference	array (one dimension) Example (array of long): [J

See <http://docs.oracle.com/javase/specs/jvms/se7/html/jvms-4.html#jvms-4.3> for more details.

WARNING:

Only primitive and their object wrapper, and strings types are supported to call a method. One dimension arrays of these types are also supported.