JBoss Tattletale 1.2 User's Guide

Betraying all your project's naughty little secrets

1

About JBoss Tattletale

JBoss Tattletale is a tool that can help development teams getting an overview of the project they are working on or a product they depend on.

The tool generates reports that will show dependencies and general information that can help identify areas that needs attention such as minimizing the number of dependencies or eliminate duplicated class files from the class path.

JBoss Tattletale will help to improve the quality of your software project.

1.1. The team

Jesper Pedersen acts as the lead for the JBoss Tattletale project. He can be reached at jesper (dot) pedersen (at) jboss (dot) org.

1.2. Thanks to

Jay Balunas, James Cobb, Torben Jaeger, Navin Surtani, Rostislav Svoboda and Steve Taranto.

2

Introduction

Have you ever found yourself frustrated with a ClassNotFoundException? Would you like to know what libraries are in your project and what they depend on? Would you like to get a full report on this stuff every time you run your ant build? If so you need to use the JBoss Tattletale project!

JBoss Tattletale is a tool that can help you get an overview of the project you are working on or a product that you depend on.

The tool will provide you with reports that can help you

- Identify dependencies between JAR files
- Find missing classes from the classpath
- Spot if a class/package is located in multiple JAR files
- Spot if the same JAR file is located in multiple locations
- With a list of what each JAR file requires and provides
- Verify the SerialVersionUID of a class
- Find similar JAR files that have different version numbers
- Find JAR files without a version number
- Find unused JAR files
- Identify sealed and signed JAR archives
- Locate a class in a JAR file
- Get the OSGi status of your project
- And generate the same reports for your .WAR and .EAR archives

JBoss Tattletale will recursive scan the directory pass as the argument for JAR files and then build the reports as HTML files.

JBoss Tattletale is licensed under GNU Lesser General Public License (LGPL) version 2.1 or later.

We hope that JBoss Tattletale will help you in your development tasks !

Please, visit the official JBoss Tattletale project page at *http://www.jboss.org/tattletale/*.

3

Getting started

3.1. Installation

JBoss Tattletale can be downloaded in its binary form for easy installation.

The download location is: *http://www.jboss.org/tattletale/downloads*

Once downloaded extract the files by executing:

```
unzip jboss-tattletale-1.2.0.Beta1.zip
```

or

```
tar xzf jboss-tattletale-1.2.0.Beta1.tar.gz
```

depending on which archive type you downloaded.

JBoss Tattletale is now located in a folder under the directory you extracted the files into.

3.1.1. Maven

The JBoss Tattletale project is published in the JBoss Maven2 repository:

```
https://repository.jboss.org/nexus/content/groups/public/org/jboss/tattletale/
```

under the group id of: org.jboss.tattletale

The artifacts for the project are

- tattletale: The core library

- tattletale-ant: The Apache Ant tasks
- tattletale-maven: The Apache Maven plugin

Developer snapshots are published on the JBoss Snapshots Maven2 server:

```
https://repository.jboss.org/nexus/content/repositories/snapshots/
```

3.1.2. Source code

If you want to experiment with the latest developments you may checkout the latest code from GitHub (formerly in SVN). Be aware that the information provided in this manual might then not be accurate.

You can clone the repository onto your computer by the following command. This automatically creates a directory 'tattletale' within your current directory. Please note that this would be a read-only directory.

```
git clone git://github.com/jesperpedersen/tattletale.git
```

Should you want to edit the sources yourself, it is best to create your own fork of the repository and then clone that particular fork on your command line. Should you need help with Git and GitHub then read the help files (http://help.github.com/)

The project is compiled using Java Development Kit 1.5 or higher, Apache Ant 1.8 or higher and Apache Ivy 2.2 or higher. Using

```
ant <target>
```

where target is one of

- dist

  Builds the distribution.

- release

  Builds the release archives.

- doc

Builds the documentation for the project.

- clean

  Cleans the project of temporary files.

See the full list of targets in the main build.xml file.

3.2. Configuration

The configuration of JBoss Tattletale is done through its

```
jboss-tattletale.properties
```

file.

The current configuration parameters includes:

Table 3.1. JBoss Tattletale configuration

Key	Value
`reports`	A comma separated list of which reports that should be generated. The following reports are supported: • `classdependants` The "Class Dependants" report. • `classdependson` The "ClassDependsOn" report. • `dependants` The "Dependants" report. • `dependson` The "DependsOn" report. • `graphviz` The "Graphical dependencies" report. • `transitivedependants` The "Transitive Dependants" report.

Key	Value
	• `transitivedependson` The "Transitive DependsOn" report. • `circulardependency` The "Circular Dependency" report. • `classlocation` The "Class Location" report. • `osgi` The "OSGi" report. • `sealed` The "Sealed Information" report. • `sign` The "Signing Information" report. • `eliminatejars` The "Eliminate Jar files with different versions" report. • `invalidversion` The "Invalid version" report. • `multiplejars` The "Multiple Jar files" report. • `multiplejarspackage` The "Multiple Jar files (Package)" report. • `multiplelocations` The "Multiple Locations" report. • `unusedjar` The "Unused Jar" report. • `blacklisted` The "Black listed" report.

Key	Value
	• `noversion` The "No version" report. • `jar` The "Jar archive" report. Default: All reports (`reports=*`)
`classloader`	Specifies which classloader structure that should be used when scanning the archives. Can be one of the following: • `org.jboss.tattletale.reporting.classloader.NoopClassLoaderStructure` A no-operation classloader structure implementation that doesn't scope any archives. • `org.jboss.tattletale.reporting.classloader.JBossAS4ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 4.x directory structures. • `org.jboss.tattletale.reporting.classloader.JBossAS5ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 5.x directory structures. • `org.jboss.tattletale.reporting.classloader.JBossAS6ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 6.x directory structures.
`profiles`	Specifies a comma separated list of profiles to resolve against. The following profiles are supported: • `java5` The Java 5 API. • `java6` The Java 6 API. • `ee5` The Java Enterprise Edition 5 API. • `ee6`

Key	Value
	The Java Enterprise Edition 6 API. • `seam22` The JBoss Seam 2.2 API. • `cdi10` The Contexts and Dependency Injection (CDI) 1.0 API. • `spring25` The Spring 2.5 API. • `spring30` The Spring 3.0 API.
`excludes`	A comma separated list of directories or files that should be excluded from the scan. F.ex. `**/server/**,myjar.jar`
`blacklisted`	A comma separated list of black listed classes or packages. F.ex. `com.mycompany.forinternaluseonly, com.partner.forinternaluseonly`
`scan`	A comma separated list of file extensions that should be scanned Default: `.jar`
`enableDot`	Should images be generated if the Graphviz DOT application is found Default: `true`

`graphvizDot`	The full path to the Graphviz DOT executable. This property is required if you want to generate PNG files and the Graphviz bin directory is not on your path. F.ex.
	`graphvizDot=C:\\Graphviz2.26.3\\bin\\dot.exe`
	or
	`graphvizDot=/opt/graphiz/bin/dot`

The load order for the configuration file is

1. `configuration` parameter in the Apache Ant / Maven task
2. `-Djboss-tattletale.properties` system property
3. `jboss-tattletale.properties` file in current directory
4. `jboss-tattletale.properties` file in class loader

NOTE: The classloader structure feature is currently based on directory structures and may therefore fail to identify archives that should be included in the reports. If you want to be sure that all archives are included use the NoopClassLoaderStructure plugin.

3.2.1. Filtering

JBoss Tattletale supports filtering of the warnings and errors that the reports generates.

This functionality allows the user to filter out any warnings or errors that should be ignored and thereby allow the application to successful terminate.

The format of the filter properties file is

Table 3.2. JBoss Tattletale filter

Key	Value
`report id`	The filter

An example:

```
myreport=myfilter
```

The load order for the filter file is

1. `filter` parameter in the Apache Ant / Maven task
2. `-Djboss-tattletale-filter.properties` system property
3. `jboss-tattletale-filter.properties` file in current directory

See the individual reports for filtering support.

3.3. Running

Running JBoss Tattletale is very easy

```
java -Xmx512m -jar tattletale.jar [-exclude=<excludes>] <sourcedir> [<outputdir>]
```

where the "sourcedir" is the directory that contains your Java archives and the optional "outputdir" parameter is the directory where you would like your reports to be generated. The "-exclude" option let you exclude directories or file on the command line - see the configuration file for syntax.

Tattletale can also be run against specific compiled Archives. These can be singular or multiple files provided in the command line and separated using a # delimeter.

```
java -Xmx512m -jar tattletale.jar myApp.jar#yourApp.war#amazingApp.ear report
```

The main file will be generated under the report directory as index.html.

JBoss Tattletale will scan for Java (.jar), Web (.war) and Enterprise (.ear) Archive files.

JBoss Tattletale requires Java Runtime Environment 5 or higher.

4

Apache Ant

JBoss Tattletale integrates with Apache Ant such that you can generate the reports directly from your build environment.

First, you need to add tattletale.jar, tattletale-ant.jar and javassist.jar to the Apache Ant classpath.

Second, you need to add the following to your project definition tag:

```
xmlns:tattletale="antlib:org.jboss.tattletale.ant"
```

That is it.

Alternative, you can do a `taskdef` for each task

```
<taskdef name="report"
         classname="org.jboss.tattletale.ant.ReportTask"
         classpathref="tattletale.lib.path.id"/>
```

See the Apache Ant documentation for additional instructions on installation.

4.1. report

Usage:

```
<tattletale:report source="${src.dir]" destination="${dest.dir}"/>
```

Table 4.1. Apache Ant: report

Key	Value
`source`	The directory that contains the Java archives. Multiple directories can be scanned by separating each with the `File.pathSeparator` character - f.ex. `dir1:dir2` on Un*x. Default: Current directory
`destination`	The directory where the reports should be generated Default: Current directory
`configuration`	Path to the configuration file Default: No value
`filter`	Path to the filter file Default: No value
`reports`	A comma separated list of which reports that should be generated. All reports can be selected by specifying "*". The following reports are supported: • `classdependants` The "Class Dependants" report. • `classdependson` The "Class DependsOn" report. • `dependants` The "Dependants" report. • `dependson` The "DependsOn" report. • `graphviz` The "Graphical dependencies" report. • `transitivedependants` The "Transitive Dependants" report. • `transitivedependson` The "Transitive DependsOn" report. • `circulardependency`

Key	Value
	The "Circular Dependency" report. • `classlocation` The "Class Location" report. • `osgi` The "OSGi" report. • `sealed` The "Sealed Information" report. • `sign` The "Signing Information" report. • `eliminatejars` The "Eliminate Jar files with different versions" report. • `invalidversion` The "Invalid version" report. • `multiplejars` The "Multiple Jar files" report. • `multiplejarspackage` The "Multiple Jar files (Package)" report. • `multiplelocations` The "Multiple Locations" report. • `unusedjar` The "Unused Jar" report. • `blacklisted` The "Black listed" report. • `noversion` The "No version" report. • `jar`

Key	Value
	The "Jar archive" report. Default: All reports
`classloader`	Specifies which classloader structure that should be used when scanning the archives. Can be one of the following: • `org.jboss.tattletale.reporting.classloader.NoopClassLoaderStructure` A no-operation classloader structure implementation that doesn't scope any archives. • `org.jboss.tattletale.reporting.classloader.JBossAS4ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 4.x directory structures. • `org.jboss.tattletale.reporting.classloader.JBossAS5ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 5.x directory structures. • `org.jboss.tattletale.reporting.classloader.JBossAS6ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 6.x directory structures. Default: `org.jboss.tattletale.reporting.classloader.NoopClassLoaderStructure`
`profiles`	Specifies a comma separated list of profiles to resolve against. All profiles can be selected by specifying "*". The following profiles are supported: • `java5` The Java 5 API. • `java6` The Java 6 API. • `ee5` The Java Enterprise Edition 5 API. • `ee6` The Java Enterprise Edition 6 API.

Key	Value
	• `seam22` The JBoss Seam 2.2 API. • `cdi10` The Contexts and Dependency Injection (CDI) 1.0 API. • `spring25` The Spring 2.5 API. • `spring30` The Spring 3.0 API. Default: `java5, java6`
`excludes`	A comma separated list of directories or files that should be excluded from the scan. F.ex. `**/server/**,myjar.jar` Default: Empty list
`blacklisted`	A comma separated list of black listed classes or packages. F.ex. `com.mycompany.forinternaluseonly, com.partner.forinternaluseonly` Default: Empty list
`failOnInfo`	Fail the build if a failed INFO report is found Default: `false`
`failOnWarn`	Fail the build if a failed WARN report is found Default: `false`
`failOnError`	Fail the build if a failed ERROR report is found Default: `false`
`deleteOutputDirectory`	Should the output directory be deleted Default: `true`
`scan`	A comma separated list of file extensions that should be scanned Default: `.jar`

Note

Note that defining a property in the task overrides the setting in the configuration file.

5

Apache Maven

JBoss Tattletale integrates with Apache Maven such that you can generate the reports directly from your build environment.

To be able to use the Tattletale Maven plugin in your Maven project, you will have to add the following plugin declaration in the pom.xml of your project:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jboss.tattletale</groupId>
      <artifactId>tattletale-maven</artifactId>
      <!-- The version of the plugin you want to use -->
      <version>1.1.0.Final</version>
      <executions>
  <execution>
    <goals>
      <goal>report</goal>
    </goals>
  </execution>
      </executions>
      <configuration>
  <!-- This is the location which will be scanned for generating tattletale reports
    -->
  <source>/absolutepath/to/source/dir</source>
  <!-- This is where the reports will be generated -->
  <destination>/absolute/path/to/reports/dir</destination>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Note

By default, the tattletale-maven plugin is attached to the "package" phase of Maven.

5.1. report

Usage:
Once you have configured your project's pom.xml to include the tattletale-maven plugin, as explained earlier, you can generate the report by running the package goal on your project

```
mvn clean package
```

Table 5.1. Apache Maven: report

Key	Value
`source`	The directory that contains the Java archives. Multiple directories can be scanned by separating each with the `File.pathSeparator` character - f.ex. `dir1:dir2` on Un*x. Default: Current directory
`destination`	The directory where the reports should be generated Default: Current directory
`configuration`	Path to the configuration file Default: No value
`filter`	Path to the filter file Default: No value
`reports`	Contains nested `report` elements of which reports that should be generated. All reports can be selected by specifying "*". The following reports are supported: • `classdependants` The "Class Dependants" report. • `classdependson` The "Class DependsOn" report. • `dependants` The "Dependants" report. • `dependson` The "DependsOn" report.

Key	Value
	• `graphviz` The "Graphical dependencies" report. • `transitivedependants` The "Transitive Dependants" report. • `transitivedependson` The "Transitive DependsOn" report. • `circulardependency` The "Circular Dependency" report. • `classlocation` The "Class Location" report. • `osgi` The "OSGi" report. • `sealed` The "Sealed Information" report. • `sign` The "Signing Information" report. • `eliminatejars` The "Eliminate Jar files with different versions" report. • `invalidversion` The "Invalid version" report. • `multiplejars` The "Multiple Jar files" report. • `multiplejarspackage` The "Multiple Jar files (Package)" report. • `multiplelocations` The "Multiple Locations" report.

Key	Value
	• `unusedjar` The "Unused Jar" report. • `blacklisted` The "Black listed" report. • `noversion` The "No version" report. • `jar` The "Jar archive" report. Default: All reports
`classloader`	Specifies which classloader structure that should be used when scanning the archives. Can be one of the following: • `org.jboss.tattletale.reporting.classloader.NoopClassLoaderStructure` A no-operation classloader structure implementation that doesn't scope any archives. • `org.jboss.tattletale.reporting.classloader.JBossAS4ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 4.x directory structures. • `org.jboss.tattletale.reporting.classloader.JBossAS5ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 5.x directory structures. • `org.jboss.tattletale.reporting.classloader.JBossAS6ClassLoaderStructure` A classloader structure implementation that scopes based on JBoss Application Server 6.x directory structures. Default: `org.jboss.tattletale.reporting.classloader.NoopClassLoaderStructure`
`profiles`	Contains nested `profile` elements of profiles to resolve against. All profiles can be selected by specifying "*". The following profiles are supported: • `java5`

Key	Value
	The Java 5 API. • `java6` The Java 6 API. • `ee5` The Java Enterprise Edition 5 API. • `ee6` The Java Enterprise Edition 6 API. • `seam22` The JBoss Seam 2.2 API. • `cdi10` The Contexts and Dependency Injection (CDI) 1.0 API. • `spring25` The Spring 2.5 API. • `spring30` The Spring 3.0 API. Default: `<report>java5</report><report>java6</report>`
`excludes`	Contains nested `exclude` elements of directories or files that should be excluded from the scan. F.ex. `<exclude>**/server/**</exclude><exclude>myjar.jar</exclude>` Default: Empty list
`blacklisted`	Contains nested `blacklist` elements of black listed classes or packages. F.ex. `<blacklist>com.mycompany.forinternaluseonly</blacklist>` `<blacklist>com.partner.forinternaluseonly</blacklist>` Default: Empty list
`failOnInfo`	Fail the build if a failed INFO report is found Default: `false`
`failOnWarn`	Fail the build if a failed WARN report is found Default: `false`

Key	Value
`failOnError`	Fail the build if a failed ERROR report is found Default: `false`
`deleteOutputDirectory`	Should the output directory be deleted Default: `true`
`scan`	A comma separated list of file extensions that should be scanned Default: `.jar`

Note

Note that defining a property in the task overrides the setting in the configuration file.

6

Reports

6.1. Dependency reports

6.1.1. Class Dependants

The class dependants report will lists which classes depends on a specific class.

Table 6.1. Class Dependants report

Class	Dependants
The class	A list of classes that depends on this class

6.1.2. Class Depends On report

The class depends on report will lists which classes that a class depends on.

Table 6.2. Class Depends On report

Class	Depends On
The class	A list of classes which the class depends on

6.1.3. Dependants

The dependants report will lists which archives depends on a specific archive.

Table 6.3. Dependants report

Archive	Dependants
The archive	A list of archives that depends on this archive

6.1.4. Depends On report

The depends on report will lists which archives that an archive depends on.

Table 6.4. Depends On report

Archive	Depends On
The archive	A list of archives which the archive depends on. Classes which can't be found are listed in italic

Filter key is: `dependson`

Filter definition is:

```
archive=[class|package](,[class|package])*;
```

An example:

```
dependson=myjar1.jar=org.eclipse.*;myjar2.jar=com.mycompany.MyClass,com.mycompany.OtherClass
```

6.1.5. Graphical dependencies report

The graphical dependencies report will create GraphViz dot files that show the dependencies as graphics.

As an example you can generate a PNG image using

```
dot -Tpng myarchive.dot > myarchive.png
```

See the GraphViz documentation for a full description on how to generate these images.

Table 6.5. Graphical dependencies report

Archive	Archives	Packages
The archive	GraphViz file that shows inter-archive dependencies	GraphViz file that shows inter-package dependencies

6.1.6. Transitive Dependants

The transitive dependants report will lists all archives depends on a specific archive.

Table 6.6. Transitive Dependants report

Archive	Dependants
The archive	A list of all archives that depends on this archive

6.1.7. Package Dependants report

The package dependants report lists the packages that depend on a specific package within your archive that you are analyzing.

Table 6.7. Package Dependants Report

Package	Dependants
The package	A list of packages that depend on this package

6.1.8. Package Dependants report

The package dependants report lists the packages that your package depends on.

Table 6.8. Package Dependants Report

Package	Depends On
The package	A list of packages that your package depends on

6.1.9. Transitive Depends On report

The transitive depends on report will lists all archives that an archive depends on.

Table 6.9. Transitive Depends On report

Archive	Depends On
The archive	A list of all archives which the archive depends on. Classes which can't be found are listed in italic

Filter key is: `transitivedependson`

Filter definition is:

```
archive=[class|package](,[class|package])*;
```

An example:

```
transitivedependson=myjar.jar=com.mycompany.MyClass,com.mycompany.OtherClass
```

```

```

6.1.10. Circular Dependency report

The circular dependency report will lists all archives that has a circular dependency with another archive.

Archives that are marked with "(*)" has the circular dependency. Note, that the circular dependency can be through a transitive dependency and not a direct dependency.

Table 6.10. Circular Dependency report

Archive	Circular Dependencies
The archive	A list of all archives which the archive has a circular dependency on.

Filter key is: `circulardependency`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
circulardependency=myjar1.jar,myjar2.jar
```

6.2. General reports

6.2.1. Class Location

The class location will lists which archives that contain a specific class file.

Table 6.11. Class Location report

Class	Jar file
The class	The list of archives that contains the class

Filter key is: `classlocation`

Filter definition is:

```

```

```
[class|package](,[class|package])*;
```

An example:

```
classlocation=org.eclipse.*,com.mycompany.MyClass
```

6.2.2. OSGi

The OSGi report will display the OSGi state of your project.

Table 6.12. OSGi report

Archive	OSGi	Report	Manifest
The archive	The OSGi state of the archive	The OSGi report for the archive	A sample OSGi enabled MANIFEST file

Filter key is: `osgi`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
osgi=myjar1.jar,myjar2.jar
```

6.2.3. Sealed information

The sealed information report will display the sealed status of your project.

Table 6.13. Sealed information report

Archive	Status
The archive	The status if the archive is sealed or unsealed

Filter key is: `sealed`

Filter definition is:

```
[yes|on|true|no|off|false]
```

An example:

```
sealed=off
```

6.2.4. Signing information

The signing information report will display the signing status of your project.

Table 6.14. Signing information report

Archive	Status
The archive	The status if the archive is signed or unsigned

Filter key is: `sign`

Filter definition is:

```
[yes|on|true|no|off|false]
```

An example:

```
sign=off
```

6.2.5. Eliminate Jar files with different versions

The eliminate jar files with different versions lists archives that have the same name but has a different version identifier.

Table 6.15. Eliminate Jar report

Archive	Location
The archive	The list of locations that the archive is found

Filter key is: `eliminatejars`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
eliminatejars=myjar1.jar,myjar2.jar
```

6.2.6. Invalid version

The invalid version report lists archives that doesn't have a valid OSGi version identifier.

Table 6.16. Invalid version report

Name	Location
The archive name	The location and version identifier for the archive

Filter key is: `invalidversion`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
invalidversion=myjar1.jar,myjar2.jar
```

6.2.7. Multiple Jar files

The multiple jar files report will list classes that appear in multiple jar files.

Table 6.17. Multiple Jar files report

Class	Jar files
The class	The list of archives where this class is found

Filter key is: `multiplejars`

Filter definition is:

```
[package](,[package])*;
```

An example:

```
multiplejars=com.mycompany.mypackage1,com.mycompany.mypackage2
```

6.2.8. Multiple Jar files (Package)

The multiple jar files fpr packages report will list packages that appear in multiple jar files.

Table 6.18. Multiple Jar files report (Package)

Package	Jar files
The package name	The list of archives where this package is found

Filter key is: `multiplejarspackage`

Filter definition is:

```
[package](,[package])*;
```

An example:

```
multiplejarspackage=com.mycompany.mypackage1,com.mycompany.mypackage2
```

6.2.9. Multiple Locations

The multiple locations report will list archives that appear in multiple locations under the scanned source directory.

Table 6.19. Multiple Locations report

Name	Location
The archive name	The list of locations where the archive is found

Filter key is: `multiplelocations`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
multiplelocations=myjar1.jar,myjar2.jar
```

6.2.10. Unused Jar

The Unused Jar report lists archives that isn't referenced from any other Jar archive in the distribution. This doesn't mean however that the archive isn't used since it could be referenced through Java Reflection or through metadata.

Table 6.20. Unused Jar report

Archive	Used
The archive	Status if the archive is used or not

Filter key is: `unusedjar`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
unusedjar=myjar1.jar,myjar2.jar
```

6.2.11. Black listed

The black listed report will list archives that uses black listed APIs.

Table 6.21. Black listed report

Archive	Usage
The archive name	The list of packages that uses black listed APIs

Filter key is: `blacklisted`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
blacklisted=myjar1.jar,myjar2.jar
```

6.2.12. No version

The no version report will list archives that doesn't have a version identifier.

Table 6.22. No version report

Name	Location
The archive name	The list of locations where the archive is found

Filter key is: `noversion`

Filter definition is:

```
[archive](,[archive])*;
```

An example:

```
noversion=myjar1.jar,myjar2.jar
```

6.2.13. JBoss AS7

The JBoss AS7 report will list the `jboss-deployment-structure.xml` file for archives.

Table 6.23. JBoss AS7 report

Name	File
The archive name	The XML file

6.3. Archive reports

6.3.1. Java ARchive (JAR)

The Java ARchive (JAR) report will provide you with an overview of the archive.

Table 6.24. No version report

Key	Value
Name	The archive name
Class Version	The version identifier for the class files
Locations	The list of locations for the archive
Manifest	The manifest file
Signing information	The signing information for the archive
Requires	The list of required classes
Provides	The list of provided classes - including SerialVersionUID (if present)

6.3.2. Web ARchive (WAR)

The Web ARchive (WAR) report will provide you with an overview of the archive, like the JAR report.

6.3.3. Java Enterprise ARchive (EAR)

The Java Enterprise ARchive (EAR) report will provide you with an overview of the archive, like the JAR report.

6.4. Custom Reports

As a user, you can define your own custom report types to tattletale. However, there are some restrictions to this and there is some required configuration. These are as follows:

- Your custom report class has to be an implementation of org.jboss.tattletale.reporting.Report
- You have to define your required reports in jboss-tattletale.properties. Each key string has to be of the form 'customreport.{int}' where int has to start with 1 and increase by 1 with each extra custom report that is required.

  For example

```
customreport.1=my.custom.package.customreport
```

- At runtime, as long as your custom jar file is in the classpath your custom report will be generated. If you were running from the command line for example:

```
java -classpath tattletale.jar:customreporting.jar org.jboss.tattletale.Main [directory-of-source-jar] [output-directory]
```

7

Troubleshooting

7.1. JBoss Tattletale generates empty reports

JBoss Tattletale generates its reports based on Java archives and not source code. Make sure that sourcedir you specify when running JBoss Tattletale contains the Java archives (f.ex. .JAR files) that you need scanned.

7.2. JBoss Tattletale throws an OutOfMemoryException

JBoss Tattletale needs to process the information it gathers in memory, so you need to provide enough memory for that to happen. You can ajust the -Xmx parameter of the command line below if you are using Sun's Java Runtime Environment.

```
java -Xmx1024m -jar jboss-tattletale.jar <sourcedir> [<outputdir>]
```

7.3. I would like to implement a feature

So you have found an area where you are missing a feature and would like to submit a patch for it, great !

There are a couple of steps to get a feature included

First, you should create a new thread in our *development forum* [http://community.jboss.org/en/tattletale/dev] where you describe the feature, its design and implementation.

Once there is an agreement on the feature and the design you should proceed with creating the patch.

To maximize your chances of getting the feature in the official build as soon as possible make sure that you run `checkstyle` to make sure that the code is correctly formatted:

```
ant clean checkstyle
```

when there are no errors, create a JIRA task (Feature) in our *JIRA environment* [https://issues.jboss.org/browse/TTALE] and attach the patch. See the developer guide for additional details.

Happy Coding !

7.4. How do I ?

We can't cover every single issue in this guide, so feel free to drop by our forums to see if a solution has already been provided. Otherwise feel free to ask your question there.

Our forum is located at *http://community.jboss.org/en/tattletale*.