Migration Toolkit for Runtimes 1.0

Maven Plugin Guide

Integrate the Migration Toolkit for Runtimes into the Maven build process.

Last Updated: 2023-03-16

Migration Toolkit for Runtimes 1.0 Maven Plugin Guide

Integrate the Migration Toolkit for Runtimes into the Maven build process.

Abstract

This guide describes how to use the Migration Toolkit for Runtimes Maven plugin to simplify migration of Java applications.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message .

CHAPTER 1. INTRODUCTION

1.1. ABOUT THE MAVEN PLUGIN GUIDE

This guide is for engineers, consultants, and others who want to use the Migration Toolkit for Runtimes (MTR) to migrate Java applications or other components. It describes how to install and run the Maven plugin, review the generated reports, and take advantage of additional features.

1.2. ABOUT THE MIGRATION TOOLKIT FOR RUNTIMES

What is the Migration Toolkit for Runtimes?
The Migration Toolkit for Runtimes (MTR) is an extensible and customizable rule-based tool that simplifies the migration and modernization of Java applications.

MTR examines application artifacts, including project source directories and application archives, and then produces an HTML report highlighting areas needing changes. MTR supports many migration paths including the following examples:

- Upgrading to the latest release of Red Hat JBoss Enterprise Application Platform
- Migrating from Oracle WebLogic or IBM WebSphere Application Server to Red Hat JBoss Enterprise Application Platform
- Containerizing applications and making them cloud-ready
- Migrating from Java Spring Boot to Quarkus
- Updating from Oracle JDK to OpenJDK
- Upgrading from OpenJDK 8 to OpenJDK 11
- Upgrading from OpenJDK11 to OpenJDK 17
- Migrating EAP Java applicatons to Azure
- Migrating Spring Boot Java applications to Azure

For more information about use cases and migration paths, see the MTR for developers web page.

How does the Migration Toolkit for Runtimes simplify migration?
The Migration Toolkit for Runtimes looks for common resources and known trouble spots when migrating applications. It provides a high-level view of the technologies used by the application.

MTR generates a detailed report evaluating a migration or modernization path. This report can help you to estimate the effort required for large-scale projects and to reduce the work involved.

How do I learn more?
See the Introduction to the Migration Toolkit for Runtimes to learn more about the features, supported configurations, system requirements, and available tools in the Migration Toolkit for Runtimes.

1.3. ABOUT THE MAVEN PLUGIN

The Maven plugin for the Migration Toolkit for Runtimes integrates into the Maven build process, allowing developers to continuously evaluate migration and modernization efforts with each iteration of source code. It provides numerous reports that highlight the analysis results, and is designed for developers who want updates with each build.

CHAPTER 2. GETTING STARTED

2.1. RUNNING THE MAVEN PLUGIN

The Maven plugin is run by including a reference to the plugin inside your application's **pom.xml** file. When the application is built, the Maven plugin is run and generates the reports for analysis.

Prerequisites

- Java Development Kit (JDK) installed. MTR supports the following JDKs:
  - OpenJDK 11
  - Oracle JDK 11
- 8 GB RAM
- macOS installation: the value of **maxproc** must be **2048** or greater.

Procedure

1. Add the following **<plugin>** to your application's **pom.xml** file:

```
[...]
<plugin>
    <groupId>org.jboss.windup.plugin</groupId>
    <artifactId>mtr-maven-plugin</artifactId>
    <version>1.0.2.GA-redhat-00001</version>
    <executions>
        <execution>
            <id>run-windup</id>
            <phase>package</phase>
            <goals>
                <goal>windup</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <target>eap:7</target> 1
    </configuration>
</plugin>
[...]
```

1 Specify a migration target. At least one migration target must be supplied within the configuration.

2. Add **--add-modules=java.se** to the **MAVEN_OPTS** environment variable.

```
export MAVEN_OPTS=--add-modules=java.se
```

3. Build the project:

```
$ mvn clean install
```

You can access the generated reports.

2.2. RUNNING THE MAVEN PLUGIN WITH MULTIPLE MODULES

To use the Maven plugin in a project with multiple modules, place the configuration inside the parent's **pom.xml**. During execution the Maven plugin will generate a single report that contains the analysis for the parent and any child modules.

NOTE

It is strongly recommended to set **inherited** to false in multi-module projects; otherwise, the Maven plugin will run when each child is compiled, resulting in multiple executions of the Maven plugin against the child modules. Setting **inherited** to false results in each project being analyzed a single time and drastically decreased run times.

To run the Maven plugin in a project with multiple modules perform the following steps.

1. Include the following plugin inside the parent project's **pom.xml**. The following is a sample **pom.xml** for a parent module.

```
<plugin>
  <groupId>org.jboss.windup.plugin</groupId>
  <artifactId>mtr-maven-plugin</artifactId>
  <version>1.0.2.GA-redhat-00001</version>
  <inherited>false</inherited>
  <executions>
    <execution>
      <id>run-windup</id>
      <phase>package</phase>
      <goals>
        <goal>windup</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <input>${project.basedir}</input>
    <target>eap:7</target> 1
    <windupHome>>/PATH/TO/CLI/<</windupHome>
  </configuration>
</plugin>
```

1 Specify a migration target. At least one migration target must be supplied within the configuration.

This **pom.xml** file differs from the default in the following attributes:

- **inherited**: Defined at the plugin level, this attribute indicates whether or not this configuration should be used in child modules. Set to **false** for performance improvements.
- **input**: Specifies the path to the directory containing the projects to be analyzed. This attribute defaults to **{project.basedir}/src/main**, and should be defined if the parent project does not have source code to analyze.

- **windupHome**: A path to an extracted copy of the MTR CLI. This attribute is optional, but is recommended as a performance improvement.
  The above example demonstrates a set of recommended arguments.

2. Build the parent project. During the build process the Maven plugin runs against all children in the project without further configuration.

   ```
   $ mvn clean install
   ```

3. Once completed, you can access the generated reports. This report contains the analysis for the parent and all children.

2.3. ACCESSING THE REPORT

When you run Migration Toolkit for Runtimes, the report is generated in the **OUTPUT_REPORT_DIRECTORY** that you specify using the **outputDirectory** argument in the **pom.xml**. Upon completion of the build, you will see the following message in the build log.

```
Windup report created: <OUTPUT_REPORT_DIRECTORY>/index.html
```

The output directory contains the following files and subdirectories:

```
<OUTPUT_REPORT_DIRECTORY>/
├── index.html          // Landing page for the report
├── <EXPORT_FILE>.csv     // Optional export of data in CSV format
├── graph/              // Generated graphs used for indexing
├── reports/            // Generated HTML reports
├── stats/              // Performance statistics
```

See the Reviewing the reports section of the MTR *CLI Guide* for information on the MTR reports and how to use them to assess your migration or modernization effort.

CHAPTER 3. EXPORTING THE REPORT IN CSV FORMAT

MTR provides the ability to export the report data, including the classifications and hints, to a flat file on your local file system. The export function currently supports the CSV file format, which presents the report data as fields separated by commas (**,**).

The CSV file can be imported and manipulated by spreadsheet software such as Microsoft Excel, OpenOffice Calc, or LibreOffice Calc. Spreadsheet software provides the ability to sort, analyze, evaluate, and manage the result data from an MTR report.

3.1. EXPORTING THE REPORT

To export the report as a CSV file, run MTR with the **exportCSV** argument set to **true**.

A CSV file is created in the directory specified by the **--output** argument for each application analyzed. All discovered issues, spanning all the analyzed applications, are included in **AllIssues.csv** file.

The CSV files are exported to the directory specified by the **outputDirectory** argument.

3.2. IMPORTING THE CSV FILE INTO A SPREADSHEET PROGRAM

1. Launch the spreadsheet software, for example, Microsoft Excel.
2. Choose **File → Open**.
3. Browse to the CSV exported file and select it.
4. The data is now ready to analyze in the spreadsheet software.

3.3. ABOUT THE CSV DATA STRUCTURE

The CSV formatted output file contains the following data fields:

Rule Id

The ID of the rule that generated the given item.

Problem type

hint or *classification*

Title

The title of the *classification* or *hint*. This field summarizes the issue for the given item.

Description

The detailed description of the issue for the given item.

Links

URLs that provide additional information about the issue. A link consists of two attributes: the link and a description of the link.

Application

The name of the application for which this item was generated.

File Name

The name of the file for the given item.

File Path

The file path for the given item.

Line

The line number of the file for the given item.

Story points

The number of story points, which represent the level of effort, assigned to the given item.

APPENDIX A. REFERENCE MATERIAL

A.1. ABOUT MAVEN PLUGIN ARGUMENTS

The following is a detailed description of the available MTR Maven plugin arguments.

Table A.1. MTR Maven plugin arguments

Argument	Description
analyzeKnownLibraries	Flag to analyze known software artifacts embedded within your application. By default, MTR only analyzes application code. **NOTE** This option may result in a longer execution time and a large number of migration issues being reported.
customLoggingPropertiesFile	An absolute path to a **logging.properties** file that contains a **java.util.logging.LogManager** logging configuration. If the specified path is invalid, or the option is not specified, then the logging reverts to using the **logging.properties** file included with the Maven plugin.
disableTattletale	Flag to disable generation of the Tattletale report. If both **enableTattletale** and **disableTattletale** are set to true, then **disableTattletale** will be ignored and the Tattletale report will still be generated.
enableCompatibleFilesReport	Flag to enable generation of the Compatible Files report. Due to processing all files without found issues, this report may take a long time for large applications.
enableTattletale	Flag to enable generation of a Tattletale report for each application. This option is enabled by default when **eap** is in the included target. If both **enableTattletale** and **disableTattletale** are set to true, then**disableTattletale** will be ignored and the Tattletale report will still be generated.

Argument	Description
enableTransactionAnalysis	[Technology Preview] Flag to enable generation of a Transactions report that displays the call stack, which executes operations on relational database tables. The Enable Transaction Analysis feature supports Spring Data JPA and the traditional **preparedStatement()** method for SQL statement execution. It does not support ORM frameworks such as Hibernate. NOTE enableTransactionAnalysis is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.
excludePackages	A list of packages to exclude from evaluation. For example, entering "com.mycompany.commonutilities" would exclude all classes whose package name begins with "com.mycompany.commonutilities".
excludeTags	A list of tags to exclude. When specified, rules with these tags will not be processed.
explodedApps	Flag to indicate that the provided input directory contains source files for a single application.
exportCSV	Flag to export the report data to a CSV file on your local file system. MTR creates the file in the directory specified by the **outputDirectory** argument. The CSV file can be imported into a spreadsheet program for data manipulation and analysis.
includeTags	A list of tags to use. When specified, only rules with these tags will be processed.
inputDirectory	Specify the path to the directory containing the applications to be analyzed. This argument defaults to **{project.basedir}/src/main/**.

Argument	Description
keepWorkDirs	Flag to instruct MTR to not delete temporary working files, such as the graph database and unzipped archives. This is useful for debugging purposes.
packages	A list of the packages to be evaluated by MTR. This argument is required.
offlineMode	Flag to operate in offline mode, disabling network access features, such as scheme validation. Used to improve performance.
outputDirectory	Specify the path to the directory to output the report information generated by MTR. This argument defaults to **{project.build.directory}/windup-report**.
overwrite	Flag to force delete the existing output directory specified by **outputDirectory**. Defaults to **true**. WARNING Be careful not to specify a report output directory that contains important information!
sourceTechnologies	A list of one or more source technologies, servers, platforms, or frameworks to migrate from. This argument, in conjunction with the **targetTechnologies** argument, helps to determine which rulesets are used.
sourceMode	Flag to indicate that the application to be evaluated contains source files rather than compiled binaries. Defaults to **true**.
targetTechnologies	A list of one or more target technologies, servers, platforms, or frameworks to migrate to. This argument, in conjunction with the **sourceTechnologies** argument, helps to determine which rulesets are used.
userIgnorePath	Specify a location for MTR to identify files that should be ignored.

Argument	Description
userRulesDirectory	Specify a location for MTR to look for custom MTR rules. The value can be a directory containing ruleset files or a single ruleset file. The ruleset files must use the **.windup.xml** suffix.
windupHome	An optional argument that points to the root of an extracted MTR CLI. By referencing a local installation of the CLI, the Maven plugin has direct access to all of the indexes, resulting in a performance increase.
windupVersion	Specify the version of MTR to run. By default, this is the Maven plugin's build version.

A.1.1. Specifying the input directory

A path to the file or directory containing one or more applications to be analyzed. This defaults to **{project.basedir}/src/main/**.

Usage

```
<inputDirectory>
    <INPUT_ARCHIVE_OR_DIRECTORY>
</inputDirectory>
```

A.1.2. Evaluating an input file

Depending on whether the input file type provided to the **inputDirectory** argument is a file or directory, it will be evaluated as follows depending on the additional arguments provided.

Directory

--explodedApp	--sourceMode	Neither Argument
The directory is evaluated as a single application.	The directory is evaluated as a single application.	Each subdirectory is evaluated as an application.

File

--explodedApp	--sourceMode	Neither Argument
Argument is ignored; the file is evaluated as a single application.	The file is evaluated as a compressed project.	The file is evaluated as a single application.

A.1.3. Specifying the output directory

Specify the path to the directory to output the report information generated by MTR.

Usage

```
<outputDirectory>
   <OUTPUT_REPORT_DIRECTORY>
</outputDirectory>
```

- If omitted, the report will be generated in the **{project.build.directory}/windup-report** directory.
- If the output directory exists, it will be overwritten based on the value of the **overwrite** argument. This argument defaults to **true**, and causes MTR to delete and recreate the directory.

A.1.4. Setting the source technology

A list of one or more source technologies, servers, platforms, or frameworks to migrate from. This argument, in conjunction with the **targetTechnologies** argument, helps to determine which rulesets are used.

Usage

```
<sourceTechnologies>
   <source>eap:6</source>
</sourceTechnologies>
```

The **sourceTechnologies** argument now provides version support, which follows the Maven version range syntax. This instructs MTR to only run the rulesets matching the specified versions. For example, **<source>eap:5</source>**.

A.1.5. Setting the target argument

A list of one or more target technologies, servers, platforms, or frameworks to migrate to. This argument, in conjunction with the **sourceTechnologies** argument, helps to determine which rulesets are used. This argument is required

Usage

```
<targetTechnologies>
  <target>eap:7</target>
</targetTechnologies>
```

The **targetTechnologies** argument now provides version support, which follows the Maven version range syntax. This instructs MTR to only run the rulesets matching the specified versions. For example, **<target>eap:7</target>**.

> **WARNING**
>
> When migrating to JBoss EAP, be sure to specify the version in the target, for example, **eap:6**. Specifying only **eap** will run rulesets for all versions of JBoss EAP, including those not relevant to your migration path.
>
> See Supported migration paths in *Introduction to the Migration Toolkit for Runtimes* for the appropriate JBoss EAP version.

A.1.6. Selecting packages

A list of the packages to be evaluated by MTR. It is highly recommended to use this argument.

Usage

```
<packages>
  <package>
    <PACKAGE_1>
  </package>
  <package>
    <PACKAGE_2>
  </package>
</packages>
```

- In most cases, you are interested only in evaluating custom application class packages and not standard Java EE or third party packages. The **<PACKAGE_N>** argument is a package prefix; all subpackages will be scanned. For example, to scan the packages **com.mycustomapp** and **com.myotherapp**, use the following snippet in your **pom.xml**.

  ```
  <packages>
    <package>com.mycustomapp</package>
    <package>com.myotherapp</package>
  </packages>
  ```

- While you can provide package names for standard Java EE third party software like **org.apache**, it is usually best not to include them as they should not impact the migration effort.

A.2. DEFAULT LOGGING PROPERTIES

The default **logging.properties** file included with the Maven plugin is provided below. This configuration omits many extraneous messages while allowing you to view the progress of the Maven plugin.

Default logging.properties file

```
# Licensed under the Eclipse Public License version 1.0, available at
# http://www.eclipse.org/legal/epl-v10.html
#

# Additional loggers to configure (the root logger is always configured)
#loggers=
```

```
handlers=java.util.logging.ConsoleHandler
.level=INFO
#java.util.logging.ConsoleHandler.level=INFO

#loggers=org.jboss.forge,org.jboss.weld,org.xnio,org.jboss.forge,org.ocpsoft.rewrite,org.jboss.windup.gr
aph.GraphModelScanner,org.jboss.windup.reporting.xml.ClassificationHandler,org.jboss.windup.graph.
GraphTyp$
org.jboss.forge.level=SEVERE
org.janusgraph.level=SEVERE
org.janusgraph.diskstorage.berkeleyje.BerkeleyJEKeyValueStore.level=SEVERE
org.janusgraph.diskstorage.berkeleyje.level=SEVERE
org.jboss.weld.level=SEVERE
org.xnio.level=SEVERE
org.jboss.forge.level=SEVERE
org.ocpsoft.rewrite.level=SEVERE
org.jboss.windup.graph.GraphModelScanner.level=SEVERE
org.jboss.windup.reporting.xml.ClassificationHandler.level=SEVERE
org.jboss.windup.graph.GraphTypeManager.level=SEVERE
org.jboss.windup.graph.GraphContextImpl.level=SEVERE
org.jboss.windup.rules.files.FileMapping.level=SEVERE
org.jboss.windup.exec.level=SEVERE
org.jboss.windup.config.level=SEVERE
com.thinkaurelius.level=SEVERE
org.jboss.windup=INFO
```

A.3. ABOUT RULE STORY POINTS

A.3.1. What are story points?

Story points are an abstract metric commonly used in Agile software development to estimate the *level of effort* needed to implement a feature or change.

The Migration Toolkit for Runtimes uses story points to express the level of effort needed to migrate particular application constructs, and the application as a whole. It does not necessarily translate to man-hours, but the value should be consistent across tasks.

A.3.2. How story points are estimated in rules

Estimating the level of effort for the story points for a rule can be tricky. The following are the general guidelines MTR uses when estimating the level of effort required for a rule.

Level of Effort	Story Points	Description
Information	0	An informational warning with very low or no priority for migration.
Trivial	1	The migration is a trivial change or a simple library swap with no or minimal API changes.
Complex	3	The changes required for the migration task are complex, but have a documented solution.

Level of Effort	Story Points	Description
Redesign	5	The migration task requires a redesign or a complete library change, with significant API changes.
Rearchitecture	7	The migration requires a complete rearchitecture of the component or subsystem.
Unknown	13	The migration solution is not known and may need a complete rewrite.

A.3.3. Task category

In addition to the level of effort, you can categorize migration tasks to indicate the severity of the task. The following categories are used to group issues to help prioritize the migration effort.

Mandatory

The task must be completed for a successful migration. If the changes are not made, the resulting application will not build or run successfully. Examples include replacement of proprietary APIs that are not supported in the target platform.

Optional

If the migration task is not completed, the application should work, but the results may not be optimal. If the change is not made at the time of migration, it is recommended to put it on the schedule soon after your migration is completed. An example of this would be the upgrade of EJB 2.x code to EJB 3.

Potential

The task should be examined during the migration process, but there is not enough detailed information to determine if the task is mandatory for the migration to succeed. An example of this would be migrating a third-party proprietary type where there is no directly compatible type.

Information

The task is included to inform you of the existence of certain files. These may need to be examined or modified as part of the modernization effort, but changes are typically not required. An example of this would be the presence of a logging dependency or a Maven **pom.xml**.

For more information on categorizing tasks, see Using custom rule categories.

A.4. ADDITIONAL RESOURCES

A.4.1. Getting involved

To help the Migration Toolkit for Runtimes cover most application constructs and server configurations, including yours, you can help with any of the following items.

- Send an email to jboss-migration-feedback@redhat.com and let us know what MTR migration rules should cover.
- Provide example applications to test migration rules.

- Identify application components and problem areas that may be difficult to migrate.
    - Write a short description of these problem migration areas.
    - Write a brief overview describing how to solve the problem migration areas.
- Try Migration Toolkit for Runtimes on your application. Be sure to report any issues you encounter.
- Contribute to the Migration Toolkit for Runtimes rules repository.
    - Write a Migration Toolkit for Runtimes rule to identify or automate a migration process.
    - Create a test for the new rule.
    - Details are provided in the *Rules Development Guide*.
- Contribute to the project source code.
    - Create a core rule.
    - Improve MTR performance or efficiency.

Any level of involvement is greatly appreciated!

A.4.2. Resources

- MTR forums: https://developer.jboss.org/en/windup
- Jira issue tracker: https://issues.redhat.com/projects/WINDUP
- MTR mailing list: jboss-migration-feedback@redhat.com

A.4.3. Reporting issues

MTR uses Jira as its issue tracking system. If you encounter an issue executing MTR, submit a Jira issue.

Revised on 2023-03-16 15:54:33 UTC