



Migration Toolkit for Applications 4.2

Maven Plugin Guide

Integrate the Red Hat Application Migration Toolkit into the Maven build process.

Migration Toolkit for Applications 4.2 Maven Plugin Guide

Integrate the Red Hat Application Migration Toolkit into the Maven build process.

Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

CHAPTER 1. INTRODUCTION	3
1.1. ABOUT THE MAVEN PLUGIN GUIDE	3
1.2. ABOUT MIGRATION TOOLKIT FOR APPLICATIONS	3
What is Migration Toolkit for Applications?	3
How Does Migration Toolkit for Applications Simplify Migration?	3
How Do I Learn More?	3
1.3. ABOUT THE MAVEN PLUGIN	3
CHAPTER 2. GETTING STARTED	4
2.1. PREREQUISITES	4
2.2. RUN THE MAVEN PLUGIN	4
2.3. RUN THE MAVEN PLUGIN WITH MULTIPLE MODULES	5
2.4. ACCESS THE REPORT	6
CHAPTER 3. EXPORT THE REPORT IN CSV FORMAT	7
3.1. EXPORT THE REPORT	7
3.2. IMPORT THE CSV FILE INTO A SPREADSHEET PROGRAM	7
3.3. OVERVIEW OF THE CSV DATA STRUCTURE	7
APPENDIX A. REFERENCE MATERIAL	9
A.1. MAVEN PLUGIN ARGUMENTS	9
A.1.1. Specify the Input Directory	11
A.1.2. Specify the Output Directory	11
A.1.3. Set the Source Technology	12
A.1.4. Set the Target Argument	12
A.1.5. Select Packages	13
A.2. DEFAULT LOGGING.PROPERTIES IN THE MAVEN PLUGIN	13
A.3. RULE STORY POINTS	14
A.3.1. What are Story Points?	14
A.3.2. How Story Points are Estimated in Rules	14
A.3.3. Task Category	15
A.4. ADDITIONAL RESOURCES	15
A.4.1. Get Involved	15
A.4.2. Important Links	16
A.4.3. Known MTA Issues	16
A.4.4. Report Issues with MTA	16
A.4.4.1. Create a JIRA Issue	16

CHAPTER 1. INTRODUCTION

1.1. ABOUT THE MAVEN PLUGIN GUIDE

This guide is for engineers, consultants, and others who want to use Migration Toolkit for Applications (MTA) 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 MIGRATION TOOLKIT FOR APPLICATIONS

What is Migration Toolkit for Applications?

Migration Toolkit for Applications (MTA) is an extensible and customizable rule-based tool that helps simplify migration of Java applications.

MTA examines application artifacts, including project source directories and application archives, then produces an HTML report that highlights areas needing changes. MTA can be used to migrate Java applications from previous versions of *Red Hat JBoss Enterprise Application Platform* or from other containers, such as *Oracle® WebLogic Server* or *IBM® WebSphere® Application Server*.

How Does Migration Toolkit for Applications Simplify Migration?

Migration Toolkit for Applications looks for common resources and highlights technologies and known trouble spots when migrating applications. The goal is to provide a high-level view into the technologies used by the application and provide a detailed report organizations can use to estimate, document, and migrate enterprise applications to Java EE and Red Hat JBoss Enterprise Application Platform.

How Do I Learn More?

See the [Getting Started Guide](#) to learn more about the features, supported configurations, system requirements, and available tools in the Migration Toolkit for Applications.

1.3. ABOUT THE MAVEN PLUGIN

The Maven plugin for Migration Toolkit for Applications 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. PREREQUISITES

Before using the Maven plugin, verify that you meet the following prerequisites.

- Java Platform, JRE version 8+
- A minimum of 4 GB RAM; 8 GB recommended
- Maven version 3.2.5 or later



NOTE

If you are running macOS, it is recommended to set the maximum number of user processes, **maxproc**, to at least **2048**, and the maximum number of open files, **maxfiles**, to **100000**.

2.2. RUN THE MAVEN PLUGIN

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

To run the Maven plugin perform the following steps.

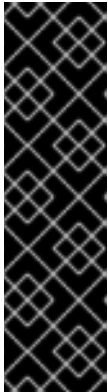
1. Include the following plugin inside your application's **pom.xml**:

```
[...]
<plugin>
  <groupId>org.jboss.windup.plugin</groupId>
  <artifactId>windup-maven-plugin</artifactId>
  <version>4.2.1.Final</version>
  <executions>
    <execution>
      <id>run-windup</id>
      <phase>package</phase>
      <goals>
        <goal>windup</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <offlineMode>true</offlineMode>
  </configuration>
</plugin>
[...]
```

- **offlineMode**: Indicates to run in offline mode, disabling network features to improve performance.
The above example demonstrates the minimum required arguments. See [MTA Maven Arguments](#) for a detailed description of all available arguments.

2. If using Java 11, then **--add-modules=java.se** must be added to the **MAVEN_OPTS** environment variable. When using older versions of Java this is not necessary, and you can proceed to the next step.

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



IMPORTANT

Using the Maven plugin on Java 11 is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See [Technology Preview Features Support Scope](#) on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

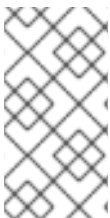
3. Build the project.

```
$ mvn clean install
```

4. [Access the generated reports](#).

2.3. RUN 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 be executed 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>windup-maven-plugin</artifactId>
  <version>4.2.1.Final</version>
  <inherited>>false</inherited>
  <executions>
    <execution>
      <id>run-windup</id>
      <phase>package</phase>
      <goals>
        <goal>windup</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

```

        </goals>
      </execution>
    </executions>
    <configuration>
      <input>${project.basedir}</input>
      <offlineMode>true</offlineMode>
      <windupHome>/PATH/TO/CLI/</windupHome>
    </configuration>
  </plugin>

```

This **pom.xml** deviates from the one in [Run the Maven Plugin](#) by 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 MTA CLI. This attribute is optional, but is recommended as a performance improvement.
- The above example demonstrates a set of recommended arguments. See [MTA Maven Arguments](#) for a detailed description of all available arguments.
- Build the parent project. During the build process the Maven plugin will execute against all children in the project without further configuration.

```
$ mvn clean install
```

- Once completed, [Access the generated reports](#) as normal. This report contains the analysis for the parent and all children.

2.4. ACCESS THE REPORT

When you execute Migration Toolkit for Applications, 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 [Review the Reports](#) section of the MTA *CLI Guide* for information on the MTA reports and how to use them to assess your migration or modernization effort.

CHAPTER 3. EXPORT THE REPORT IN CSV FORMAT

MTA 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 MTA report.

3.1. EXPORT THE REPORT

To export the report into a CSV file, run MTA with **exportCSV** argument set to **true**. A CSV file will be created in the directory specified by the **--output** argument for each application analyzed. All discovered issues, spanning all the analyzed applications, will be included in **AllIssues.csv**.

```
<exportCSV>true</exportCSV>
```

The CSV files will be created in the directory specified by the **outputDirectory** argument.

3.2. IMPORT 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.

For more information or to resolve any issues, check the help for your spreadsheet software.

3.3. OVERVIEW OF 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. MAVEN PLUGIN ARGUMENTS

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

Table A.1. MTA Maven plugin Arguments

Argument	Description
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.
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. See the Input File Argument Tables for details.
exportCSV	Flag to export the report data to a CSV file on your local file system. MTA 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. For details, see Export the Report in CSV Format .
includeTags	A list of tags to use. When specified, only rules with these tags will be processed.

Argument	Description
inputDirectory	Specify the path to the directory containing the applications to be analyzed. This argument defaults to {project.basedir}/src/main/ . See Specify the Input Directory for more information.
keepWorkDirs	Flag to instruct MTA 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 MTA. This argument is required. See Select Packages for more information.
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 MTA. This argument defaults to {project.build.directory}/windup-report . See Specify the Output Directory for more information.
overwrite	<p>Flag to force delete the existing output directory specified by outputDirectory. Defaults to true.</p> <div>  <p>WARNING</p> <p>Be careful not to specify a report output directory that contains important information!</p> </div>
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. See Set the Source Technology for more information.
sourceMode	Flag to indicate that the application to be evaluated contains source files rather than compiled binaries. Defaults to true . See the Input File Argument Tables for details.
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. See Set the Target Technology for more information.

Argument	Description
userIgnorePath	Specify a location for MTA to identify files that should be ignored.
userRulesDirectory	Specify a location for MTA to look for custom MTA rules. The value can be a directory containing ruleset files or a single ruleset file. The ruleset files must use the .windup.xml or .rhamt.xml suffix.
windupHome	An optional argument that points to the root of an extracted MTA 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 MTA to run. By default, this is the Maven plugin's build version.

A.1.1. Specify 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>
```

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.1.2. Specify the Output Directory

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

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 MTA to delete and recreate the directory.

A.1.3. Set 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 MTA to only run the rulesets matching the specified versions. For example, **<source>eap:5</source>**.

A.1.4. Set 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 MTA 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 the MTA *Getting Started Guide* for which JBoss EAP version is appropriate for your source platform.

A.1.5. Select Packages

A list of the packages to be evaluated by MTA. 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 IN THE MAVEN PLUGIN

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.

```
#
# Copyright 2013-2014 Red Hat, Inc. and/or its affiliates.
#
# 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.graph.GraphModelScanner,org.jboss.windup.reporting.xml.ClassificationHandler,org.jboss.windup.graph.GraphType$
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. 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.

Migration Toolkit for Applications 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 MTA 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.
Redesign	5	The migration task requires a redesign or a complete library change, with significant API changes.

Level of Effort	Story Points	Description
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](#) in the *Rules Development Guide*.

A.4. ADDITIONAL RESOURCES

A.4.1. Get Involved

To help make Migration Toolkit for Applications 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 MTA 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 Applications on your application. Be sure to [report any issues](#) you encounter.
- Contribute to the Migration Toolkit for Applications rules repository.
 - Write a Migration Toolkit for Applications 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 MTA performance or efficiency.
 - See the [Core Development Guide](#) for information about how to configure your environment and set up the project.

Any level of involvement is greatly appreciated!

A.4.2. Important Links

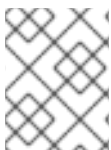
- MTA forums: <https://developer.jboss.org/en/windup>
- MTA JIRA issue trackers
 - Core MTA: <https://issues.jboss.org/browse/WINDUP>
 - MTA Rules: <https://issues.jboss.org/browse/WINDUPRULE>
- MTA mailing list: jboss-migration-feedback@redhat.com
- MTA on Twitter: [@JBossWindup](#)
- MTA IRC channel: Server FreeNode ([irc.freenode.net](#)), channel **#windup** ([transcripts](#)).

A.4.3. Known MTA Issues

You can review known issues for MTA here: [Open MTA issues](#) .

A.4.4. Report Issues with MTA

Migration Toolkit for Applications uses JIRA as its issue tracking system. If you encounter an issue executing MTA, please file a JIRA Issue.



NOTE

If you do not have one already, you must sign up for a JIRA account in order to create a JIRA issue.

A.4.4.1. Create a JIRA Issue

1. Open a browser and navigate to the JIRA [Create Issue](#) page.
If you have not yet logged in, click the **Log In** link at the top right side of the page and enter your credentials.
2. Choose the following options and click the **Next** button.
 - **Project**
For core MTA issues, choose *Red Hat Application Migration Toolkit (WINDUP)*.

For issues with MTA rules, choose: *Red Hat Application Migration Toolkit rules (WINDUPRULE)*.
 - **Issue Type:** *Bug*
3. On the next screen complete the following fields.
 - **Summary:** Enter a brief description of the problem or issue.
 - **Environment:** Provide the details of your operating system, version of Java, and any other pertinent information.
 - **Description:** Provide a detailed description of the issue. Be sure to include logs and exceptions traces.
 - **Attachment:** If the application or archive causing the issue does not contain sensitive information and you are comfortable sharing it with the MTA development team, attach it to the issue using the **browse** button.
4. Click the **Create** button to create the JIRA issue.

Revised on 2022-11-10 16:18:10 UTC