



Migration Toolkit for Runtimes 1.2

CLI Guide

Learn how to use the Migration Toolkit for Runtimes CLI to migrate your applications.

Migration Toolkit for Runtimes 1.2 CLI Guide

Learn how to use the Migration Toolkit for Runtimes CLI to migrate your applications.

Legal Notice

Copyright © 2024 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 Runtimes CLI to simplify migration of Java applications.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
CHAPTER 1. INTRODUCTION	5
1.1. ABOUT THE CLI GUIDE	5
1.2. ABOUT THE MIGRATION TOOLKIT FOR RUNTIMES	5
What is the Migration Toolkit for Runtimes?	5
How does the Migration Toolkit for Runtimes simplify migration?	5
How do I learn more?	5
1.3. THE MTR CLI	5
CHAPTER 2. INSTALLING AND RUNNING THE CLI	7
2.1. INSTALLING THE CLI	7
2.2. RUNNING THE CLI	7
2.2.1. MTR command examples	8
Running MTR on an application archive	8
Running MTR on source code	8
Running cloud-readiness rules	8
Overriding MTR properties	8
2.2.2. MTR CLI Bash completion	8
Enabling Bash completion	9
Enabling persistent Bash completion	9
2.2.3. Accessing MTR help	9
2.2.4. Using OpenRewrite recipes	9
2.2.4.1. Available OpenRewrite recipes	10
2.3. ACCESSING REPORTS	11
CHAPTER 3. REVIEWING THE REPORTS	12
3.1. APPLICATION REPORT	13
3.1.1. Dashboard	13
3.1.2. Issues report	15
3.1.3. Application details report	16
3.1.4. Technologies report	17
3.1.5. Transactions report	18
3.1.6. Source report	18
3.2. TECHNOLOGIES REPORT	19
3.3. ARCHIVES SHARED BY MULTIPLE APPLICATIONS	19
3.4. RULE PROVIDERS EXECUTION OVERVIEW	20
CHAPTER 4. EXPORTING THE REPORT IN CSV FORMAT	21
4.1. EXPORTING THE REPORT	21
Accessing the report from the application report	21
4.2. IMPORTING THE CSV FILE INTO A SPREADSHEET PROGRAM	21
4.3. ABOUT THE CSV DATA STRUCTURE	21
CHAPTER 5. MAVENIZING YOUR APPLICATION	23
5.1. GENERATING THE MAVEN PROJECT STRUCTURE	23
5.2. REVIEWING THE MAVEN PROJECT STRUCTURE	23
Root POM file	24
BOM file	24
Application POM files	24
CHAPTER 6. OPTIMIZING MTR PERFORMANCE	26
6.1. DEPLOYING AND RUNNING THE APPLICATION	26

6.2. UPGRADING HARDWARE	26
6.3. CONFIGURING MTR TO EXCLUDE PACKAGES AND FILES	26
6.3.1. Excluding packages	26
6.3.2. Excluding files	27
6.3.3. Searching locations for exclusion	27
APPENDIX A. REFERENCE MATERIAL	28
A.1. ABOUT MTR COMMAND-LINE ARGUMENTS	28
A.1.1. Specifying the input	32
A.1.2. Specifying the output directory	33
A.1.3. Setting the source technology	33
A.1.4. Setting the target technology	34
A.1.5. Selecting packages	34
A.2. SUPPORTED TECHNOLOGY TAGS	35
A.3. ABOUT RULE STORY POINTS	48
A.3.1. What are story points?	48
A.3.2. How story points are estimated in rules	48
A.3.3. Task category	49
A.4. ADDITIONAL RESOURCES	49
A.4.1. Contributing to the project	49
A.4.2. Migration Toolkit for Runtimes development resources	50
A.4.3. Reporting issues	50

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 CLI 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 CLI, 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 OpenJDK 11 to OpenJDK 17
- Upgrading from OpenJDK 17 to OpenJDK 21
- Migrating EAP Java applications 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. THE MTR CLI

The CLI is a command-line tool in the Migration Toolkit for Runtimes that you can use to assess and

prioritize migration and modernization efforts for applications. It provides numerous reports that highlight the analysis without using the other tools. The CLI includes a wide array of customization options. By using the CLI, you can tune MTR analysis options or integrate with external automation tools.

CHAPTER 2. INSTALLING AND RUNNING THE CLI

2.1. INSTALLING THE CLI

You can install the CLI on Linux, Windows, or macOS operating systems.

Prerequisites

The following are the prerequisites for the Migration Toolkit for Runtimes (MTR) installation:

- Java Development Kit (JDK) is installed. MTR supports the following JDKs:
 - OpenJDK 11
 - OpenJDK 17
 - Oracle JDK 11
 - Oracle JDK 17
 - Eclipse Temurin™ JDK 11
 - Eclipse Temurin™ JDK 17
- 8 GB RAM
- macOS installation: the value of **maxproc** must be **2048** or greater.

Procedure

1. Navigate to the [MTR Download page](#) and download the **Migration Toolkit CLI** file.
2. Extract the **.zip** file to a directory of your choice.
When you encounter **<MTR_HOME>** in this guide, replace it with the actual path to your MTR installation.

2.2. RUNNING THE CLI

You can run MTR against your application.

Procedure

1. Open a terminal and navigate to the **<MTR_HOME>/bin/** directory.
2. Execute the **windup-cli** script, or **windup-cli.bat** for Windows, and specify the appropriate arguments:

```
$ ./windup-cli --input /path/to/jee-example-app-1.0.0.ear \  
  --output /path/to/output --source weblogic --target eap:6 \  
  --packages com.acme org.apache
```

- **--input**: The application to be evaluated.
- **--output**: The output directory for the generated reports.

- **--source:** The source technology for the application migration.
- **--target:** The target technology for the application migration.
- **--packages:** The packages to be evaluated. This argument is highly recommended to improve performance.

3. Access the report.

2.2.1. MTR command examples

Running MTR on an application archive

The following command analyzes the **com.acme** and **org.apache** packages of the [jee-example-app-1.0.0.ear](#) example EAR archive for migrating from JBoss EAP 5 to JBoss EAP 7:

```
$ <MTR_HOME>/bin/windup-cli \
  --input /path/to/jee-example-app-1.0.0.ear \
  --output /path/to/report-output/ --source eap:5 --target eap:7 \
  --packages com.acme org.apache
```

Running MTR on source code

The following command analyzes the **org.jboss.seam** packages of the [seam-booking-5.2](#) example source code for migrating to JBoss EAP 6.

```
$ <MTR_HOME>/bin/windup-cli --sourceMode --input /path/to/seam-booking-5.2/ \
  --output /path/to/report-output/ --target eap:6 --packages org.jboss.seam
```

Running cloud-readiness rules

The following command analyzes the **com.acme** and **org.apache** packages of the [jee-example-app-1.0.0.ear](#) example EAR archive for migrating to JBoss EAP 7. It also evaluates for cloud readiness:

```
$ <MTR_HOME>/windup-cli --input /path/to/jee-example-app-1.0.0.ear \
  --output /path/to/report-output/ \
  --target eap:7 --target cloud-readiness --packages com.acme org.apache
```

Overriding MTR properties

To override the default *Fernflower* decompiler, pass the **-Dwindup.decompiler** argument on the command line. For example, to use the *Procyon* decompiler, use the following syntax:

```
$ <MTR_HOME>/bin/windup-cli -Dwindup.decompiler=procyon \
  --input <INPUT_ARCHIVE_OR_DIRECTORY> --output <OUTPUT_REPORT_DIRECTORY> \
  --target <TARGET_TECHNOLOGY> --packages <PACKAGE_1> <PACKAGE_2>
```

2.2.2. MTR CLI Bash completion

The MTR CLI provides an option to enable Bash completion for Linux systems, allowing the MTR command-line arguments to be auto completed by pressing the Tab key when entering the commands. For instance, when Bash completion is enabled, entering the following displays a list of available arguments.

```
$ <MTR_HOME>/bin/windup-cli [TAB]
```

Enabling Bash completion

To enable Bash completion for the current shell, execute the following command:

```
$ source <MTR_HOME>/bash-completion/windup-cli
```

Enabling persistent Bash completion

The following commands allow Bash completion to persist across restarts:

- To enable Bash completion for a specific user across system restarts, include the following line in that user's `~/.bashrc` file.

```
source <MTR_HOME>/bash-completion/windup-cli
```

- To enable Bash completion for all users across system restarts, copy the Migration Toolkit for Runtimes CLI Bash completion file to the `/etc/bash_completion.d/` directory as the root user.

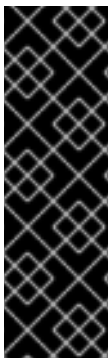
```
# cp <MTR_HOME>/bash-completion/windup-cli /etc/bash_completion.d/
```

2.2.3. Accessing MTR help

To see the complete list of available arguments for the `windup-cli` command, open a terminal, navigate to the `<MTR_HOME>` directory, and execute the following command:

```
$ <MTR_HOME>/bin/windup-cli --help
```

2.2.4. Using OpenRewrite recipes



IMPORTANT

OpenRewrite recipe support 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.

You can refactor the source code of Java applications by using [OpenRewrite](#) recipes with the MTR CLI.

For example, the OpenRewrite recipe `org.jboss.windup.JavaxToJakarta` renames imported `javax` packages to their `jakarta` equivalents.

Procedure

1. Run `windup-cli`, specifying the recipe name, the path to the configuration file, and the application:

```
$ ./windup-cli --openrewrite --input </path/to/source/project> \
  "-Drewrite.configLocation=<path/to/rewrite.yaml>" \
  "-DactiveRecipes=<recipe_name>" --goal dryRun
```

- **"-DactiveRecipes=<recipe name>":** Specify the OpenRewrite recipe, for example, **org.jboss.windup.JavaxToJakarta**.
- **--input:** Specify the application to be refactored. The application must be the top of the source code project containing a Maven Project Object Model (POM) XML file, **pom.xml**.
- **-Drewrite.configLocation=<path/to/rewrite.yaml> :** The location of the **rewrite.yaml** configuration file to use. The shipped **rewrite.yaml** configuration files are located in your **<MTR_HOME>/rules/openrewrite** subfolder, for example, **"-Drewrite.configLocation=<MTR_HOME>/rules/openrewrite/jakarta/javax/imports/rewrite.yaml"**.
- **"-DactiveRecipes=<recipe name>":** Specify the OpenRewrite recipe, for example, **org.jboss.windup.JavaxToJakarta**.
You can include more than one recipe by specifying each in the **activeRecipes** parameter. For example, to include the recipes **org.jboss.windup.JavaxInjectToJakartaInject** and **org.jboss.windup.JavaxEjbToJakartaEjb**", enter the following for **"-DactiveRecipes=<recipe name>":**

```
"-DactiveRecipes=org.jboss.windup.JavaxInjectToJakartaInject, \
  org.jboss.windup.JavaxEjbToJakartaEjb"
```

- **--goal:** Optional: The OpenRewrite Maven goal to run.
 - **dryRun** : The script returns a list of proposed changes. Ignore the **"Run 'mvn rewrite:run' to apply the recipes"** message.
 - **run**: The script applies the changes.

2. Run **windup-cli** with **--goal run** to apply the recipe:

```
$ ./windup-cli --openrewrite --input </path/to/source/project> \
  "-Drewrite.configLocation=<path/to/rewrite.yaml>" \
  "-DactiveRecipes=<recipe_name>" --goal run
```

2.2.4.1. Available OpenRewrite recipes

Table 2.1. Available OpenRewrite recipes

Migration path	Purpose	rewrite.configLocation	activeRecipes
Java EE to Jakarta EE	<p>Replace import of javax packages with equivalent jakarta packages</p> <p>Replace javax artifacts, declared within pom.xml files, with the jakarta equivalents</p>	<MTR_HOME>/rules/openrewrite/jakarta \ /javax/imports/rewrite .yaml	org.jboss.windup.JavaxToJakarta

Migration path	Purpose	rewrite.configLocation	activeRecipes
Java EE to Jakarta EE	Rename bootstrapping files	<code><MTR_HOME>/rules/opensslrewrite/jakarta \ /javax/bootstrapping/rewrite.yml</code>	<code>org.jboss.windup.java \ rta.javax. \ BootstrappingFiles</code>
Java EE to Jakarta EE	Transform persistence.xml configuration	<code><MTR_HOME>/rules/opensslrewrite/jakarta \ /javax/xml/rewrite.yml</code>	<code>org.jboss.windup.java \ x-jakarta. \ PersistenceXML</code>
Spring Boot to Quarkus	Replace spring.jpa.hibernate.ddl-auto property within files matching application*.properties	<code><MTR_HOME>/rules/opensslrewrite/quarkus \ /springboot/properties/rewrite.yml</code>	<code>org.jboss.windup.sb-quarkus.Properties</code>

2.3. ACCESSING REPORTS

When you run the Migration Toolkit for Runtimes, a report is generated in the `<OUTPUT_REPORT_DIRECTORY>` that you specify using the `--output` argument in the command line.

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
├── archives/          // Archives extracted from the application
├── mavenized/         // Optional Maven project structure
├── reports/           // Generated HTML reports
└── stats/             // Performance statistics

```

Procedure

1. Obtain the path of the **index.html** file of your report from the output that appears after you run MTR:

```

Report created: <OUTPUT_REPORT_DIRECTORY>/index.html
Access it at this URL: file:///<OUTPUT_REPORT_DIRECTORY>/index.html

```

2. Open the **index.html** file by using a browser. The generated report is displayed.

CHAPTER 3. REVIEWING THE REPORTS

The report examples shown in the following sections are a result of analyzing the **com.acme** and **org.apache** packages in the [jee-example-app-1.0.0.ear](#) example application, which is located in the MTR GitHub source repository.

The report was generated using the following command.

```
$ <MTR_HOME>/mta-cli --input /home/username/mta-cli-source/test-files/jee-example-app-1.0.0.ear/
--output /home/username/mta-cli-reports/jee-example-app-1.0.0.ear-report --target eap:6 --packages
com.acme org.apache
```

Use a browser to open the **index.html** file located in the report output directory. This opens a landing page that lists the applications that were processed. Each row contains a high-level overview of the story points, number of incidents, and technologies encountered in that application.

Figure 3.1. Application list

The screenshot shows the 'Applications' landing page. At the top, there's a search bar and a 'Tag' dropdown. Below that is a table with the following columns: Name, Runtime labels, Tags, Incidents, and Story points. The first row shows the application 'jee-example-app-1.0.0.ear' with runtime labels 'Unsuitable Tomcat' and 'Unsuitable Wildfly', 14 incidents, and 151 story points. Below the table, there are various technology tags: Apache Log4J, EAR, EJB XML 2.1, JSF XML 1.2, JTA, Manifest, Maven XML, Message (MDB), Properties, Servlet, Stateless (SLSB), Web XML 2.4, WebLogic EJB XML, and WebLogic Web XML. The page also includes pagination controls at the bottom right.



NOTE

The incidents and estimated story points change as new rules are added to MTR. The values here may not match what you see when you test this application.

The following table lists all of the reports and pages that can be accessed from this main MTR landing page. Click the name of the application, **jee-example-app-1.0.0.ear**, to view the application report.

Page	How to Access
Application	Click the name of the application.
Technologies report	Click the Technologies link at the top of the page.
Archives shared by multiple applications	Click the Archives shared by multiple applications link. Note that this link is only available when there are shared archives across multiple applications.
Rule providers execution overview	Click the Rule providers execution overview link at the bottom of the page.

Note that if an application shares archives with other analyzed applications, you will see a breakdown of how many story points are from shared archives and how many are unique to this application.

Figure 3.2. Shared archives

Application: Archives shared by multip... ▾

Issues
This report provides a concise summary of all issues identified.

Search: Category ▾ Level effort ▾ Source ▾ Target ▾ [Export CSV](#) 1 - 2 of 2 < >

Issue	Category	Source technolo...	Target technolo...	Level of effort	Total incidents	Total storypoi...
> Embedded framework - AOP Alliance	information			Info	1	0
> Embedded library - Apache Commons Logging	information			Info	1	0

1 - 2 of 2 << < 1 of 1 > >>

Information about the archives that are shared among applications can be found in the Archives Shared by Multiple Applications reports.

3.1. APPLICATION REPORT

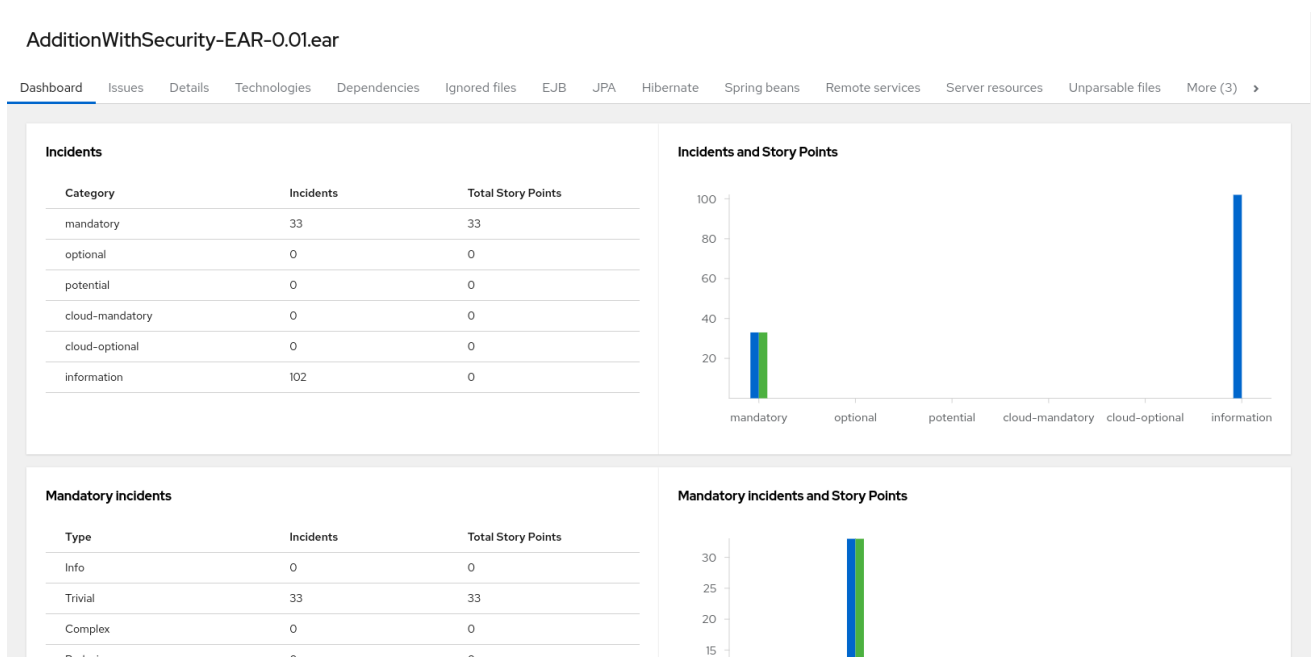
3.1.1. Dashboard

Access this report from the report landing page by clicking on the application name in the **Application List**.

The dashboard gives an overview of the entire application migration effort. It summarizes:

- The incidents and story points by category
- The incidents and story points by level of effort of the suggested changes
- The incidents by package

Figure 3.3. Dashboard



The top navigation bar lists the various reports that contain additional details about the migration of this application. Note that only those reports that are applicable to the current application will be available.

Report	Description
Issues	Provides a concise summary of all issues that require attention.
Application details	Provides a detailed overview of all resources found within the application that may need attention during the migration.
Technologies	Displays all embedded libraries grouped by functionality, allowing you to quickly view the technologies used in each application.
Dependencies	Displays all Java-packaged dependencies found within the application.
Unparsable	Shows all files that MTR could not parse in the expected format. For instance, a file with a .xml or .wsdl suffix is assumed to be an XML file. If the XML parser fails, the issue is reported here and also where the individual file is listed.
Remote services	Displays all remote services references that were found within the application.
EJBs	Contains a list of EJBs found within the application.
JBPM	Contains all of the JBPM-related resources that were discovered during analysis.
JPA	Contains details on all JPA-related resources that were found in the application.

Report	Description
Hibernate	Contains details on all Hibernate-related resources that were found in the application.
Server resources	Displays all server resources (for example, JNDI resources) in the input application.
Spring Beans	Contains a list of Spring Beans found during the analysis.
Hard-coded IP addresses	Provides a list of all hard-coded IP addresses that were found in the application.
Ignored files	Lists the files found in the application that, based on certain rules and MTR configuration, were not processed. See the --userIgnorePath option for more information.
About	Describes the current version of MTR and provides helpful links for further assistance.

3.1.2. Issues report

Access this report from the dashboard by clicking the **Issues** link.

This report includes details about every issue that was raised by the selected migration paths. The following information is provided for each issue encountered:

- A title to summarize the issue.
- The total number of incidents, or times the issue was encountered.
- The rule story points to resolve a single instance of the issue.
- The estimated level of effort to resolve the issue.
- The total story points to resolve every instance encountered. This is calculated by multiplying the number of incidents found by the story points per incident.

Figure 3.4. Issues report

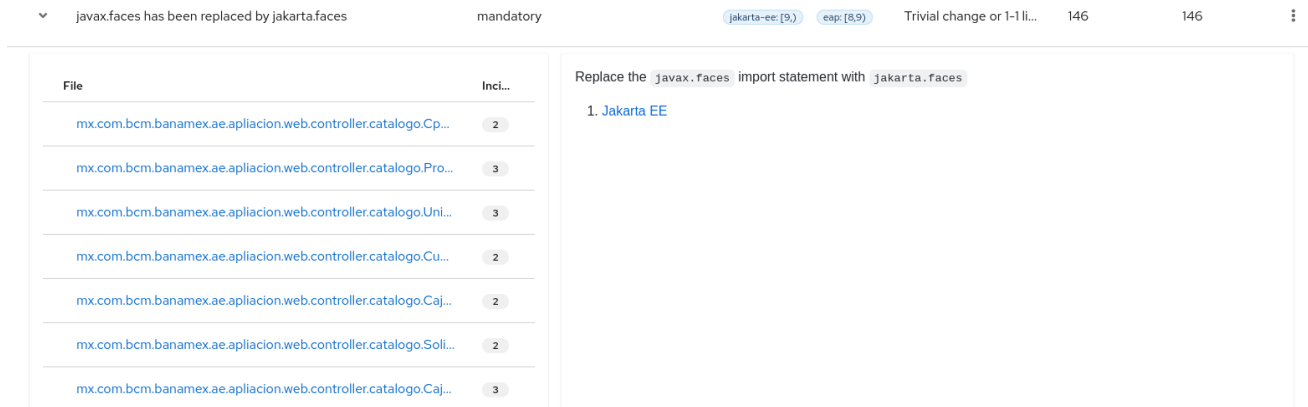
The screenshot shows the 'Issues' report interface. At the top, there is a navigation bar with links: Dashboard, Issues (selected), Details, Technologies, Dependencies, Ignored files, EJB, JPA, Hibernate, Spring beans, Remote services, Server resources, Unparsable files, and More (3). Below the navigation bar is a search and filter section with a search input, dropdowns for Category, Level effort, Source, and Target, and an 'Export CSV' button. The main content is a table with the following columns: Issue, Category, Source technolo..., Target technolo..., Level of effort, Total incidents, and Total storypoi... The table contains three rows of data:

Issue	Category	Source technolo...	Target technolo...	Level of effort	Total incidents	Total storypoi...
> Common Annotations	information			Info	1	0
> Embedded framework - AOP Alliance	information			Info	1	0
> Embedded framework - Apache Aries	information			Info	4	0

Each reported issue may be expanded, by clicking on the title, to obtain additional details. The following information is provided.

- A list of files where the incidents occurred, along with the number of incidents within each file. If the file is a Java source file, then clicking the filename will direct you to the corresponding Source report.
- A detailed description of the issue. This description outlines the problem, provides any known solutions, and references supporting documentation regarding either the issue or resolution.
- A direct link, entitled **Show Rule**, to the rule that generated the issue.

Figure 3.5. Expanded issue



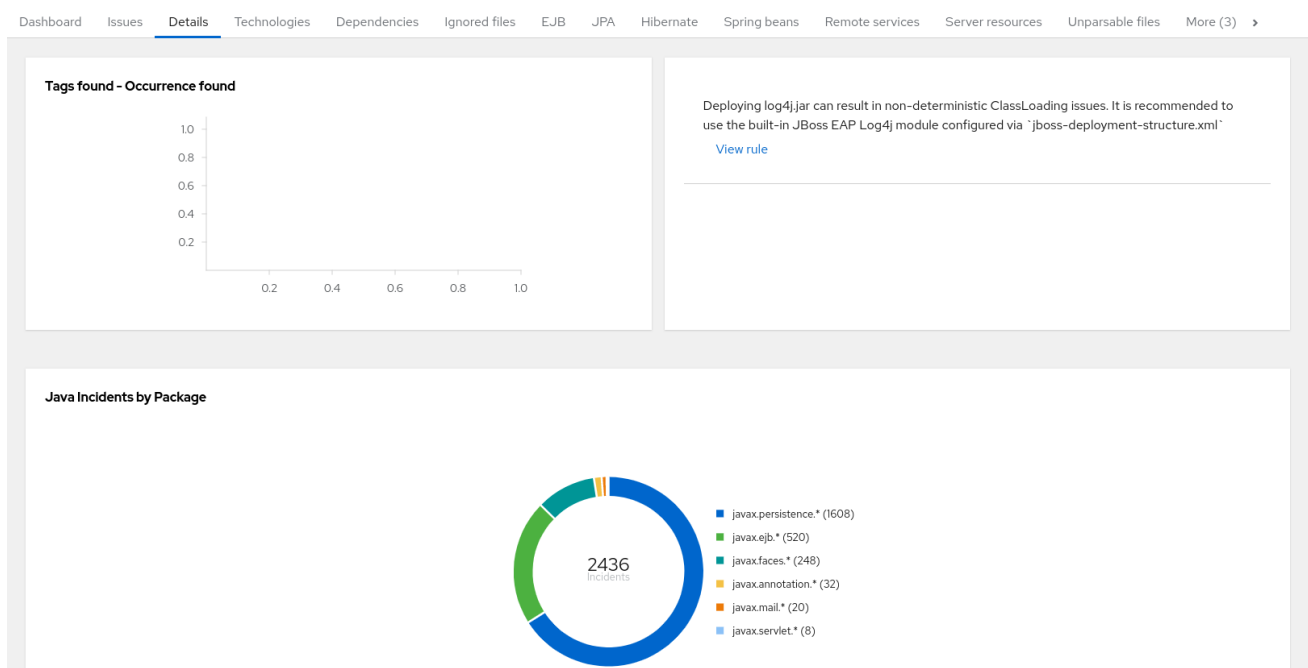
Issues are sorted into four categories by default. Information on these categories is available at ask Category.

3.1.3. Application details report

Access this report from the dashboard by clicking the **Application Details** link.

The report lists the story points, the Java incidents by package, and a count of the occurrences of the technologies found in the application. Next is a display of application messages generated during the migration process. Finally, there is a breakdown of this information for each archive analyzed during the process.

Figure 3.6. Application Details report

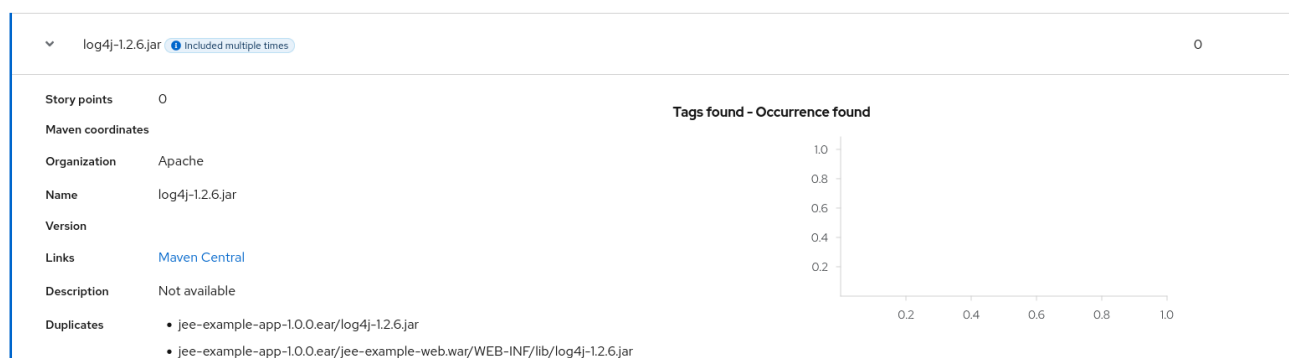


Expand the **jee-example-app-1.0.0.ear/jee-example-services.jar** to review the story points, Java incidents by package, and a count of the occurrences of the technologies found in this archive. This summary begins with a total of the story points assigned to its migration, followed by a table detailing the changes required for each file in the archive. The report contains the following columns.

Column Name	Description
Name	The name of the file being analyzed.
Technology	The type of file being analyzed, for example, Decompiled Java File or Properties .
Issues	Warnings about areas of code that need review or changes.
Story Points	Level of effort required to migrate the file.

Note that if an archive is duplicated several times in an application, it will be listed just once in the report and will be tagged with **[Included multiple times]**.

Figure 3.7. Duplicate archive in an application



The story points for archives that are duplicated within an application will be counted only once in the total story point count for that application.

3.1.4. Technologies report

Access this report from the dashboard by clicking the **Technologies** link.

The report lists the occurrences of technologies, grouped by function, in the analyzed application. It is an overview of the technologies found in the application, and is designed to assist users in quickly understanding each application's purpose.

The image below shows the technologies used in the **jee-example-app**.

Figure 3.8. Technologies in an application

Dashboard Issues Details **Technologies** Dependencies Ignored files EJB JPA Hibernate Spring beans Remote services Server resources Unparsable files More (3) >

Category ▼

Category	Technology	Count
EJB Connect	Stateless (SLSB)	75
HTTP Connect	Servlet	2
Other Connect	Mail	26
	EAR Deployment	1
	Properties	5
	Common Annotations	1
Inversion of Control Execute	AOP Alliance	2
	Spring	11
	Spring DI	2
Processing Execute	Spring Scheduled	1
	Java EE XML	1
	Quartz	4
Database Store	JDBC	2
	JDBC XA datasources	1

3.1.5. Transactions report

A Transactions report 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.

The image below shows an example of a Transactions report.

Figure 3.9. Transactions report

Dashboard Issues Details Technologies Dependencies Ignored files EJB JPA Hibernate Spring beans Remote services Server resources Unparsable files **Transactions** >

1 - 10 of 14 < >

Entry class	Entry method
> org.springframework.samples.petclinic.owner.PetController	findOwner
> org.springframework.samples.petclinic.owner.PetController	populatePetTypes
> org.springframework.samples.petclinic.owner.OwnerController	processFindForm
> org.springframework.samples.petclinic.owner.VisitController	loadPetWithVisit
> org.springframework.samples.petclinic.owner.OwnerController	processUpdateOwnerForm
> org.springframework.samples.petclinic.owner.OwnerController	initUpdateOwnerForm
> org.springframework.samples.petclinic.owner.PetController	processUpdateForm

3.1.6. Source report

The Source report displays the migration issues in the context of the source file in which they were discovered.

Figure 3.10. Source report

File AdministracionEfectivo.ear/AdministracionEfectivo-web-0.0.1-SNAPSHOT.war/WEB-INF/classes/mx/com/bcm/banamex/ae/apliacion/web/controller/catalogo/... x

```

5 import javax.faces.bean.ManagedBean;
6 import javax.faces.bean.RequestScoped;
7 import mx.com.bcm.banamex.ae.negocio.facade.CatalogoFacade;
8 import mx.com.bcm.banamex.ae.persistencia.exception.EfectivoAplicacionB0Exception;
9 import mx.com.bcm.banamex.ae.persistencia.vo.CajaVO;
10
11 @ManagedBean(
12     name = "cajaMB"
13 )
14 @RequestScoped
15 public class CajaMB implements Serializable {
16     private static final long serialVersionUID = 1L;
17     @EJB
18     private CatalogoFacade catalogoFacade;
19     private CajaVO cajaVO = new CajaVO();
20
21     public void consultCajas() throws EfectivoAplicacionB0Exception {
22     }
23
24     public void ConsultaEditCajas() throws EfectivoAplicacionB0Exception {
25     }
26
27     public void findByIpAddressCajaIdnTipoCaja(String cajaIpAddress, short cajaIdn, short cajaTipo
28     )
29     }
30     public void addCaja(CajaVO cajaVO) throws EfectivoAplicacionB0Exception {
31     }
32
33     public void formatoIp(CajaVO cajaVO) throws EfectivoAplicacionB0Exception {
34     }
35     }

```

Annotation `javax.faces.bean.RequestScoped` removed x
Line: 6
Annotation `javax.faces.bean.RequestScoped` removed. Use `jakarta.enterprise.context.RequestScoped` to replace it.

Close

3.2. TECHNOLOGIES REPORT

Access this report from the report landing page by clicking the **Technologies** link.

This report provides an aggregate listing of the technologies used, grouped by function, for the analyzed applications. It shows how the technologies are distributed, and is typically reviewed after analyzing a large number of applications to group the applications and identify patterns. It also shows the size, number of libraries, and story point totals of each application.

Clicking any of the headers, such as **Markup**, sorts the results in descending order. Selecting the same header again will resort the results in ascending order. The currently selected header is identified in bold, next to a directional arrow, indicating the direction of the sort.

Figure 3.11. Technologies used across multiple applications

Application:

Dependencies

This report is a statistic of technologies occurrences in the input applications. It shows how the technologies are distributed and is mostly useful when analysing many applications.

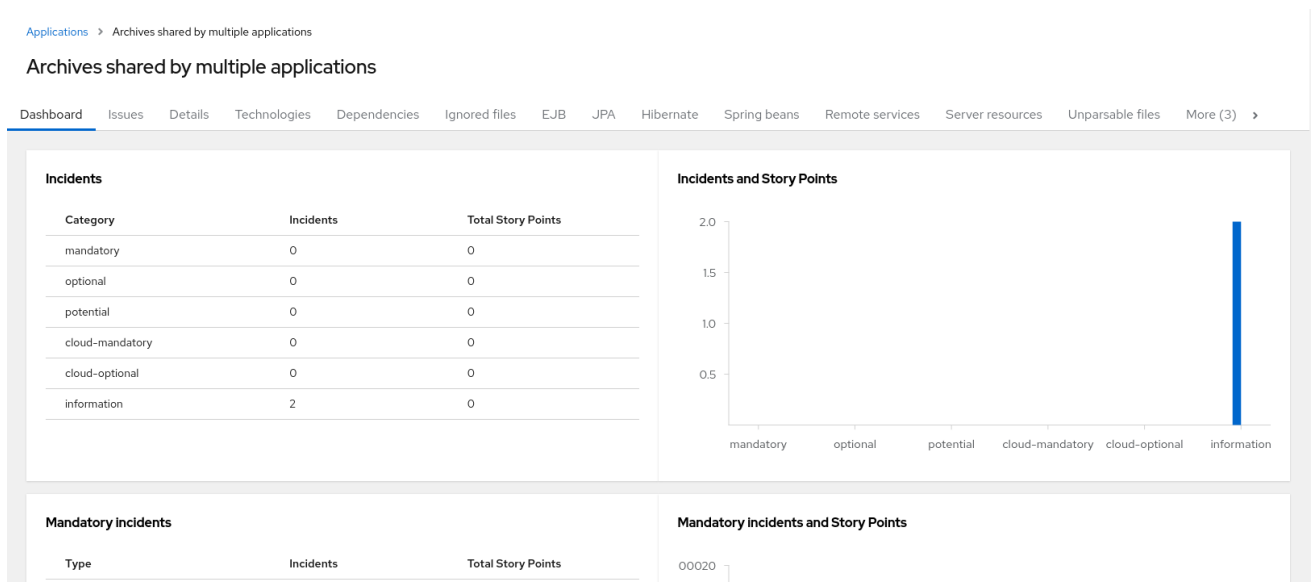
Application	Markup	MVC	Rich	Web
AdditionWithSecurity-EAR-0.01.ear	0	3	0	4
AdministracionEfectivo.ear	127	4	0	140
Archives shared by multiple applications	0	0	0	0

1 - 3 of 3 < >
1 - 3 of 3 << >> 1 of 1 >>>

3.3. ARCHIVES SHARED BY MULTIPLE APPLICATIONS

Access these reports from the report landing page by clicking the **Archives shared by multiple applications** link. Note that this link is only available if there are applicable shared archives.

Figure 3.12. Archives shared by multiple applications



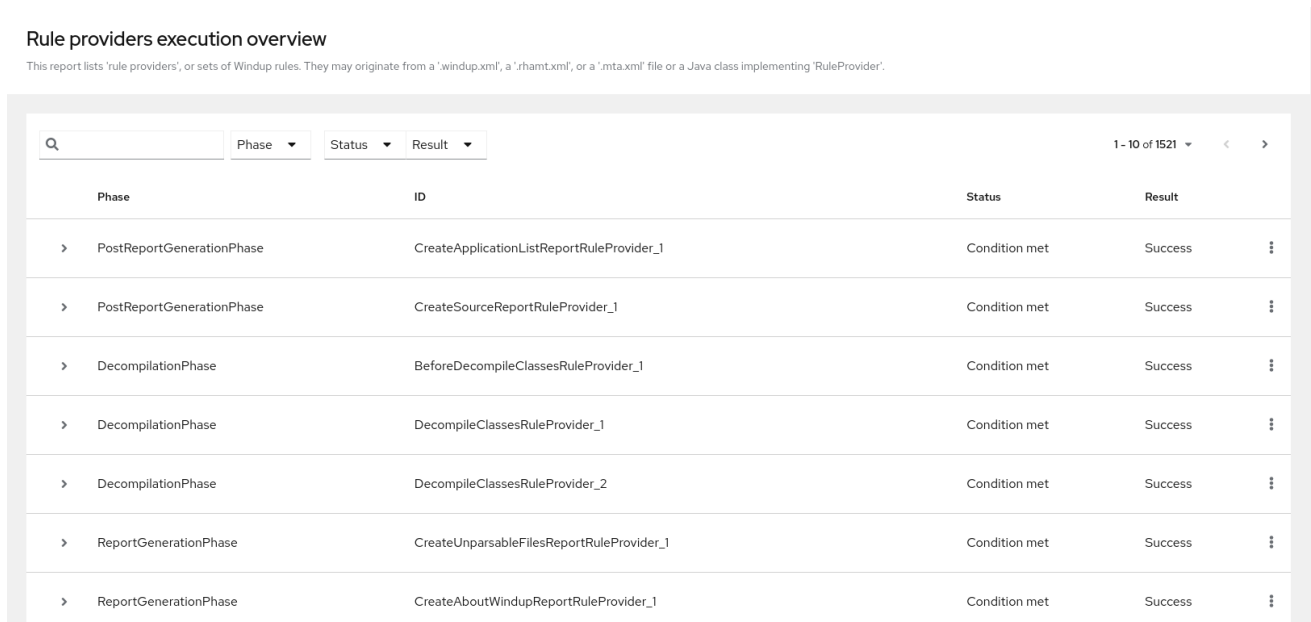
This allows you to view the detailed reports for all archives that are shared across multiple applications.

3.4. RULE PROVIDERS EXECUTION OVERVIEW

Access this report from the report landing page by clicking the **Rule providers execution overview** link.

This report provides the list of rules that ran when running the MTR migration command against the application.

Figure 3.13. Rule providers execution overview



CHAPTER 4. EXPORTING THE REPORT IN CSV FORMAT

Migration Toolkit for Runtimes (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.

4.1. EXPORTING THE REPORT

To export the report as a CSV file, run MTR with the **--exportCSV** argument. 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 the **AllIssues.csv** file that is exported to the root directory of the report.

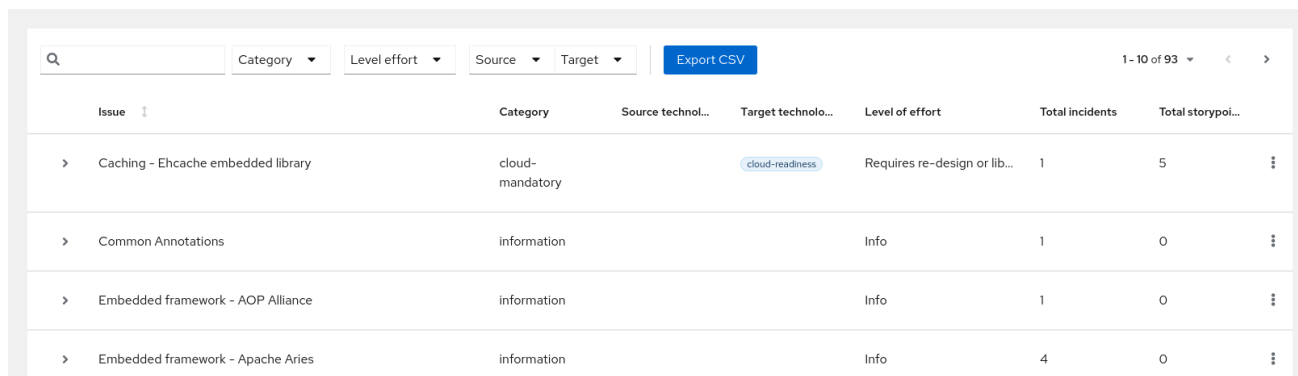
Accessing the report from the application report

If you have exported the CSV report, you can download all of the CSV issues in the Issues Report. To download these issues, click **Download All Issues CSV** in the Issues Report.

Figure 4.1. Issues report with CSV download

Issues

This report provides a concise summary of all issues identified.



Issue	Category	Source technol...	Target technolo...	Level of effort	Total incidents	Total storypoi...
> Caching - Ehcache embedded library	cloud-mandatory		cloud-readiness	Requires re-design or lib...	1	5
> Common Annotations	information			Info	1	0
> Embedded framework - AOP Alliance	information			Info	1	0
> Embedded framework - Apache Aries	information			Info	4	0

4.2. 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.

4.3. 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.

CHAPTER 5. MAVENIZING YOUR APPLICATION

MTR provides the ability to generate an Apache Maven project structure based on the application provided. This will create a directory structure with the necessary Maven Project Object Model (POM) files that specify the appropriate dependencies.

Note that this feature is not intended to create a final solution for your project. It is meant to give you a starting point and identify the necessary dependencies and APIs for your application. Your project may require further customization.

5.1. GENERATING THE MAVEN PROJECT STRUCTURE

You can generate a Maven project structure for the provided application by passing in the **--mavenize** flag when executing MTR.

The following example runs MTR using the [jee-example-app-1.0.0.ear](#) test application:

```
$ <MTR_HOME>/mta-cli --input /path/to/jee-example-app-1.0.0.ear --output /path/to/output --target
eap:6 --packages com.acme org.apache --mavenize
```

This generates the Maven project structure in the **/path/to/output/mavenized** directory.



NOTE

You can only use the **--mavenize** option when providing a compiled application for the **--input** argument. This feature is not available when running MTR against source code.

You can also use the **--mavenizeGroupId** option to specify the **<groupId>** to be used for the POM files. If unspecified, MTR will attempt to identify an appropriate **<groupId>** for the application, or will default to **com.mycompany.mavenized**.

5.2. REVIEWING THE MAVEN PROJECT STRUCTURE

The **/path/to/output/mavenized/<APPLICATION_NAME>/** directory contains the following items:

- A root **POM** file. This is the **pom.xml** file at the top-level directory.
- A BOM file. This is the **POM** file in the directory ending with **-bom**.
- One or more application **POM** files. Each module has its **POM** file in a directory named after the archive.

The example **jee-example-app-1.0.0.ear** application is an EAR archive that contains a WAR and several JARs. There is a separate directory created for each of these artifacts. Below is the Maven project structure created for this application.

```
/path/to/output/mavenized/jee-example-app/
  jee-example-app-bom/pom.xml
  jee-example-app-ear/pom.xml
  jee-example-services2-jar/pom.xml
  jee-example-services-jar/pom.xml
  jee-example-web-war/pom.xml
  pom.xml
```

Review each of the generated files and customize as appropriate for your project. To learn more about Maven POM files, see the [Introduction to the POM](#) section of the Apache Maven documentation.

Root POM file

The root POM file for the **jee-example-app-1.0.0.ear** application can be found at `/path/to/output/mavenized/jee-example-app/pom.xml`. This file identifies the directories for all of the project modules.

The following modules are listed in the root POM for the example **jee-example-app-1.0.0.ear** application.

```
<modules>
  <module>jee-example-app-bom</module>
  <module>jee-example-services2-jar</module>
  <module>jee-example-services-jar</module>
  <module>jee-example-web-war</module>
  <module>jee-example-app-ear</module>
</modules>
```



NOTE

Be sure to reorder the list of modules if necessary so that they are listed in an appropriate build order for your project.

The root POM is also configured to use the [Red Hat JBoss Enterprise Application Platform Maven repository](#) to download project dependencies.

BOM file

The Bill of Materials (BOM) file is generated in the directory ending in **-bom**. For the example **jee-example-app-1.0.0.ear** application, the BOM file can be found at `/path/to/output/mavenized/jee-example-app/jee-example-app-bom/pom.xml`. The purpose of this BOM is to have the versions of third-party dependencies used by the project defined in one place. For more information on using a BOM, see the [Introduction to the dependency mechanism](#) section of the Apache Maven documentation.

The following dependencies are listed in the BOM for the example **jee-example-app-1.0.0.ear** application

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.6</version>
    </dependency>
    <dependency>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
      <version>2.5</version>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Application POM files

Each application module that can be mavenized has a separate directory containing its POM file. The directory name contains the name of the archive and ends in a **-jar**, **-war**, or **-ear** suffix, depending on the archive type.

Each application POM file lists that module's dependencies, including:

- Third-party libraries
- Java EE APIs
- Application submodules

For example, the POM file for the **jee-example-app-1.0.0.ear** EAR, `/path/to/output/mavenized/jee-example-app/jee-example-app-ear/pom.xml`, lists the following dependencies.

```
<dependencies>
  <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.6</version>
  </dependency>
  <dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jee-example-web-war</artifactId>
    <version>1.0</version>
    <type>war</type>
  </dependency>
  <dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jee-example-services-jar</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jee-example-services2-jar</artifactId>
    <version>1.0</version>
  </dependency>
</dependencies>
```

CHAPTER 6. OPTIMIZING MTR PERFORMANCE

MTR performance depends on a number of factors, including hardware configuration, the number and types of files in the application, the size and number of applications to be evaluated, and whether the application contains source or compiled code. For example, a file that is larger than 10 MB may need a lot of time to process.

In general, MTR spends about 40% of the time decompiling classes, 40% of the time executing rules, and the remainder of the time processing other tasks and generating reports. This section describes what you can do to improve the performance of MTR.

6.1. DEPLOYING AND RUNNING THE APPLICATION

Try these suggestions first before upgrading hardware.

- If possible, run MTR against the source code instead of the archives. This eliminates the need to decompile additional JARs and archives.
- Specify a comma-separated list of the packages to be evaluated by MTR using the **--packages** argument on the `<MTR_HOME>/bin/mtr-cli` command line. If you omit this argument, MTR will decompile everything, which has a big impact on performance.
- Specify the **--excludeTags** argument where possible to exclude them from processing.
- Avoid decompiling and analyzing any unnecessary packages and files, such as proprietary packages or included dependencies.
- Increase your ulimit when analyzing large applications. See [this Red Hat Knowledgebase article](#) for instructions on how to do this for Red Hat Enterprise Linux.
- If you have access to a server that has better resources than your laptop or desktop machine, you may want to consider running MTR on that server.

6.2. UPGRADING HARDWARE

If the application and command-line suggestions above do not improve performance, you may need to upgrade your hardware.

- If you have access to a server that has better resources than your laptop/desktop, then you may want to consider running MTR on that server.
- Very large applications that require decompilation have large memory requirements. 8 GB RAM is recommended. This allows 3 - 4 GB RAM for use by the JVM.
- An upgrade from a single or dual-core to a quad-core CPU processor provides better performance.
- Disk space and fragmentation can impact performance. A fast disk, especially a solid-state drive (SSD), with greater than 4 GB of defragmented disk space should improve performance.

6.3. CONFIGURING MTR TO EXCLUDE PACKAGES AND FILES

6.3.1. Excluding packages

You can exclude packages during decompilation and analysis to increase performance. References to these packages remain in the application's source code but excluding them avoids the decompilation and analysis of proprietary classes.

Any packages that match the defined value are excluded. For example, you can use **com.acme** to exclude both **com.acme.example** and **com.acme.roadrunner**.

You can exclude packages by either of the following methods:

- Using the **--excludePackages** argument.
- Specifying the packages in a file contained within one of the ignored locations. Each package should be included on a separate line, and the file must end in **.package-ignore.txt**. For example, see **<MTR_HOME>/ignore/proprietary.package-ignore.txt**.

6.3.2. Excluding files

MTR can exclude specific files, such as included libraries or dependencies, during scanning and report generation. Excluded files are defined in a file with the **.mtr-ignore.txt** or **.windup-ignore.txt** extension within one of the ignored locations.

These files contain a regex string detailing the name to exclude, with one file listed per line. For example, you can exclude the library **ant.jar** and any Java source files beginning with **Example** with a file containing the following:

```
.*ant.jar  
.*Example.*\.java
```

6.3.3. Searching locations for exclusion

MTR searches the following locations:

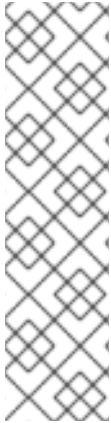
- **~/.mtr/ignore/**
- **~/.windup/ignore/**
- **<MTR_HOME>/ignore/**
- Any files and folders specified by the **--userIgnorePath** argument

Each of these files must conform to the rules specified for excluding packages or files, depending on the type of content to be excluded.

APPENDIX A. REFERENCE MATERIAL

A.1. ABOUT MTR COMMAND-LINE ARGUMENTS

The following is a detailed description of the available MTR command line arguments.




NOTE

To run the MTR command without prompting, for example when executing from a script, you must use the following arguments:


- **--batchMode**
- **--overwrite**
- **--input**
- **--target**

Table A.1. MTR CLI arguments

Argument	Description
<code>--additionalClassPath</code>	A space-delimited list of additional JAR files or directories to add to the class path so that they are available for decompilation or other analysis.
<code>--addonDir</code>	Add the specified directory as a custom add-on repository.
<code>--analyzeKnownLibraries</code>	Flag to analyze known software artifacts embedded within your application. By default, MTR only analyzes application code. <div style="margin-top: 10px;">  <p>NOTE</p> <p>This option may result in a longer execution time and a large number of migration issues being reported.</p> </div>
<code>--batchMode</code>	Flag to specify that MTR should be run in a non-interactive mode without prompting for confirmation. This mode takes the default values for any parameters not passed in to the command line.
<code>--debug</code>	Flag to run MTR in debug mode.
<code>--disableTattletale</code>	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.

Argument	Description
--discoverPackages	Flag to list all available packages in the input binary application.
--enableClassNotFoundAnalysis	Flag to enable analysis of Java files that are not available on the class path. This should not be used if some classes will be unavailable at analysis time.
--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.
--enableTransactionAnalysis	<p>[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.</p> <div data-bbox="687 1137 798 1518" style="float: left; width: 60px; height: 170px; background: repeating-linear-gradient(45deg, transparent, transparent 2px, #ccc 2px, #ccc 4px); border: 1px solid #ccc; margin-right: 10px;"></div> <p>NOTE</p> <p>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.</p>
--excludePackages	A space-delimited list of packages to exclude from evaluation. For example, entering com.mycompany.commonutilities excludes all classes whose package names begin with com.mycompany.commonutilities .
--excludeTags	A space-delimited list of tags to exclude. When specified, rules with these tags will not be processed. To see the full list of tags, use the --listTags argument.
--explodedApp	Flag to indicate that the provided input directory contains source files for a single application.

Argument	Description
<code>--exportCSV</code>	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 <code>--output</code> argument. The CSV file can be imported into a spreadsheet program for data manipulation and analysis.
<code>--exportSummary</code>	Flag to instruct MTR to generate an analysisSummary.json export file in the output directory. The file contains analysis summary information for each application analyzed, including the number of incidents and story points by category, and all of the technology tags associated with the analyzed applications.
<code>--exportZipReport</code>	This argument creates a reports.zip file in the output folder. The file contains the analysis output, typically, the reports. If requested, it can also contain the CSV export files.
<code>--help</code>	Display the MTR help message.
<code>--immutableAddonDir</code>	Add the specified directory as a custom read-only add-on repository.
<code>--includeTags</code>	A space-delimited list of tags to use. When specified, only rules with these tags will be processed. To see the full list of tags, use the <code>--listTags</code> argument.
<code>--input</code>	A space-delimited list of the path to the file or directory containing one or more applications to be analyzed. This argument is required.
<code>--install</code>	Specify add-ons to install. The syntax is <GROUP_ID>: <ARTIFACT_ID>[:<VERSION>] . For example, <code>--install core-addon-x</code> or <code>--install org.example.addon:example:1.0.0</code> .
<code>--keepWorkDirs</code>	Flag to instruct MTR to not delete temporary working files, such as the graph database and extracted archive files. This is useful for debugging purposes.
<code>--legacyReports</code>	Flag to instruct MTR to generate the old format reports instead of the new format reports.
<code>--list</code>	Flag to list installed add-ons.
<code>--listSourceTechnologies</code>	Flag to list all available source technologies.
<code>--listTags</code>	Flag to list all available tags.

Argument	Description
--listTargetTechnologies	Flag to list all available target technologies.
--mavenize	Flag to create a Maven project directory structure based on the structure and content of the application. This creates pom.xml files using the appropriate Java EE API and the correct dependencies between project modules. See also the --mavenizeGroupld option.
--mavenizeGroupld	When used with the --mavenize option, all generated pom.xml files will use the provided value for their <groupld> . If this argument is omitted, MTR will attempt to determine an appropriate <groupld> based on the application, or will default to com.mycompany.mavenized .
--online	Flag to allow network access for features that require it. Currently only validating XML schemas against external resources relies on Internet access. Note that this comes with a performance penalty.
--output	Specify the path to the directory to output the report information generated by MTR.
--overwrite	<p>Flag to force delete the existing output directory specified by --output. If you do not specify this argument and the --output directory exists, you are prompted to choose whether to overwrite the contents.</p> <div data-bbox="687 1312 794 1447" style="display: inline-block; vertical-align: middle;">  </div> <p style="margin-left: 20px;">IMPORTANT</p> <p style="margin-left: 20px;">Do not overwrite a report output directory that contains important information.</p>
--packages	A space-delimited list of the packages to be evaluated by MTR. It is highly recommended to use this argument.
--remove	Remove the specified add-ons. The syntax is <GROUP_ID>:<ARTIFACT_ID>[:<VERSION>] . For example, --remove core-addon-x or --remove org.example.addon:example:1.0.0 .
--skipReports	Flag to indicate that HTML reports should not be generated. A common use of this argument is when exporting report data to a CSV file using --exportCSV .

Argument	Description
<code>--source</code>	A space-delimited list of one or more source technologies, servers, platforms, or frameworks to migrate from. This argument, in conjunction with the --target argument, helps to determine which rulesets are used. Use the --listSourceTechnologies argument to list all available sources.
<code>--sourceMode</code>	Flag to indicate that the application to be evaluated contains source files rather than compiled binaries. The <code>sourceMode</code> argument has been deprecated. There is no longer the need to specify it. MTR can intuitively process any inputs that are presented to it. In addition, project source folders can be analyzed with binary inputs within the same analysis execution.
<code>--target</code>	A space-delimited list of one or more target technologies, servers, platforms, or frameworks to migrate to. This argument, in conjunction with the --source argument, helps to determine which rulesets are used. Use the --listTargetTechnologies argument to list all available targets.
<code>--userIgnorePath</code>	Specify a location, in addition to `\${user.home}/.mtr/ignore/` , for MTR to identify files that should be ignored.
<code>--userLabelsDirectory</code>	Specify a location for MTR to look for custom Target Runtime Labels. The value can be a directory containing label files or a single label file. The Target Runtime Label files must use the .windup.label.xml suffix. The shipped Target Runtime Labels are defined within `\${MTR_HOME}/rules/migration-core/core.windup.label.xml` .
<code>--userRulesDirectory</code>	Specify a location, in addition to `\${MTR_HOME}/rules/` and `\${user.home}/.mtr/rules/` , 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.
<code>--version</code>	Display the MTR version.
<code>--skipSourceCodeReports</code>	This option skips generating a <i>Source code report</i> when generating the application analysis report. Use this advanced option when concerned about source code information appearing in the application analysis.

A.1.1. Specifying the input

A space-delimited list of the path to the file or directory containing one or more applications to be analyzed. This argument is required.

Usage

```
--input <INPUT_ARCHIVE_OR_DIRECTORY> [...]
```

Depending on whether the input file type provided to the **--input** 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. Specifying the output directory

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

Usage

```
--output <OUTPUT_REPORT_DIRECTORY>
```

- If omitted, the report will be generated in an **<INPUT_ARCHIVE_OR_DIRECTORY>.report** directory.
- If the output directory exists, you will be prompted with the following (with a default of N).

```
Overwrite all contents of "/home/username/<OUTPUT_REPORT_DIRECTORY>" (anything already in the directory will be deleted)? [y,N]
```

However, if you specify the **--overwrite** argument, MTR will proceed to delete and recreate the directory. See the description of this argument for more information.

A.1.3. Setting the source technology

A space-delimited list of one or more source technologies, servers, platforms, or frameworks to migrate from. This argument, in conjunction with the **--target** argument, helps to determine which rulesets are used. Use the **--listSourceTechnologies** argument to list all available sources.

Usage

```
--source <SOURCE_1> <SOURCE_2>
```

The **--source** 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**.



WARNING

When migrating to JBoss EAP, be sure to specify the version, 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.1.4. Setting the target technology

A space-delimited list of one or more target technologies, servers, platforms, or frameworks to migrate to. This argument, in conjunction with the **--source** argument, helps to determine which rulesets are used. If you do not specify this option, you are prompted to select a target. Use the **--listTargetTechnologies** argument to list all available targets.

Usage

```
--target <TARGET_1> <TARGET_2>
```

The **--target** 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**.



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.1.5. Selecting packages

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

Usage

```
--packages <PACKAGE_1> <PACKAGE_2> <PACKAGE_N>
```

- 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 **--packages com.mycustomapp com.myotherapp** argument on the command line.
- 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.



WARNING

If you omit the **--packages** argument, every package in the application is scanned, which can impact performance.

A.2. SUPPORTED TECHNOLOGY TAGS

The following technology tags are supported in MTR 1.2.6:

- OMQ Client
- 3scale
- Acegi Security
- AcrIS Security
- ActiveMQ library
- Airframe
- Airlift Log Manager
- AKKA JTA
- Akka Testkit
- Amazon SQS Client
- AMQP Client
- Anakia
- AngularFaces
- ANTLR StringTemplate
- AOP Alliance
- Apache Accumulo Client
- Apache Aries

- Apache Commons JCS
- Apache Commons Validator
- Apache Flume
- Apache Geronimo
- Apache Hadoop
- Apache HBase Client
- Apache Ignite
- Apache Karaf
- Apache Mahout
- Apache Meerowave JTA
- Apache Sirona JTA
- Apache Synapse
- Apache Tapestry
- Apiman
- Applet
- Arquillian
- AspectJ
- Atomikos JTA
- Avalon Logkit
- Axion Driver
- Axis
- Axis2
- BabbageFaces
- Bean Validation
- BeanInject
- Blaze
- Blitz4j
- BootsFaces
- Bouncy Castle

- ButterFaces
- Cache API
- Cactus
- Camel
- Camel Messaging Client
- Camunda
- Cassandra Client
- CDI
- Cfg Engine
- Chunk Templates
- Cloudera
- Coherence
- Common Annotations
- Composite Logging
- Composite Logging JCL
- Concordion
- CSS
- Cucumber
- Dagger
- DbUnit
- Demoiselle JTA
- Derby Driver
- Drools
- DVSL
- Dynacache
- EAR Deployment
- Easy Rules
- EasyMock
- Eclipse RCP

- EclipseLink
- Ehcache
- EJB
- EJB XML
- Elasticsearch
- Entity Bean
- EtlUnit
- Eureka
- Everit JTA
- Evo JTA
- Feign
- File system Logging
- FormLayoutMaker
- FreeMarker
- Geronimo JTA
- GFC Logging
- GIN
- GlassFish JTA
- Google Guice
- Grails
- Grapht DI
- Guava Testing
- GWT
- H2 Driver
- Hamcrest
- Handlebars
- HavaRunner
- Hazelcast
- Hdiv

- Hibernate
- Hibernate Cfg
- Hibernate Mapping
- Hibernate OGM
- HighFaces
- HornetQ Client
- HSQLDB Driver
- HTTP Client
- HttpUnit
- ICEfaces
- Ickenham
- Ignite JTA
- Iksan
- iLog
- Infinispan
- Injekt for Kotlin
- Iroh
- Istio
- Jamon
- Jasypt
- Java EE Batch
- Java EE Batch API
- Java EE JACC
- Java EE JAXB
- Java EE JAXR
- Java EE JSON-P
- Java Transaction API
- JavaFX
- JavaScript

- Javax Inject
- JAX-RS
- JAX-WS
- JayWire
- JBehave
- JBoss Cache
- JBoss EJB XML
- JBoss logging
- JBoss Transactions
- JBoss Web XML
- JBossMQ Client
- JBPM
- JCA
- Jcabi Log
- JCache
- JUnit
- JDBC
- JDBC datasources
- JDBC XA datasources
- Jersey
- Jetbrick Template
- Jetty
- JFreeChart
- JFunk
- JGoodies
- JMock
- JMockit
- JMS Connection Factory
- JMS Queue

- JMS Topic
- JMustache
- JNA
- JNI
- JNLP
- JPA entities
- JPA Matchers
- JPA named queries
- JPA XML
- JSecurity
- JSF
- JSF Page
- JSilver
- JSON-B
- JSP Page
- JSTL
- JTA
- Jukito
- JUnit
- Ka DI
- Keyczar
- Kibana
- KLogger
- Kodein
- Kotlin Logging
- Koinject
- KumuluzEE JTA
- LevelDB Client
- Liferay

- LiferayFaces
- Lift JTA
- Log.io
- Log4J
- Log4s
- Logback
- Logging Utils
- Logstash
- Lumberjack
- Macros
- Magicgrouplayout
- Mail
- Management EJB
- MapR
- MckoiSQLDB Driver
- Memcached
- Message (MDB)
- Micro DI
- Micrometer
- Microsoft SQL Driver
- MiGLayout
- MinLog
- Mixer
- Mockito
- MongoDB Client
- Monolog
- Morphia
- MRules
- Mule

- Mule Functional Test Framework
- MultithreadedTC
- Mycontainer JTA
- MyFaces
- MySQL Driver
- Narayana Arjuna
- Needle
- Neo4j
- NLOG4J
- Nuxeo JTA/JCA
- OACC
- OAUTH
- OCPsoft Logging Utils
- OmniFaces
- OpenFaces
- OpenPojo
- OpenSAML
- OpenWS
- OPS4J Pax Logging Service
- Oracle ADF
- Oracle DB Driver
- Oracle Forms
- Orion EJB XML
- Orion Web XML
- Oscache
- OTR4J
- OW2 JTA
- OW2 Log Util
- OWASP CSRF Guard

- OWASP ESAPI
- Peaberry
- Pega
- Persistence units
- Petals EIP
- PicketBox
- PicketLink
- PicoContainer
- Play
- Play Test
- Plexus Container
- Polyforms DI
- Portlet
- PostgreSQL Driver
- PowerMock
- PrimeFaces
- Properties
- Qpid Client
- RabbitMQ Client
- RandomizedTesting Runner
- Resource Adapter
- REST Assured
- Restito
- RichFaces
- RMI
- RocketMQ Client
- Rythm Template Engine
- SAML
- Santuario

- Scalate
- Scaldi
- Scribe
- Seam
- Security Realm
- ServiceMix
- Servlet
- ShiftOne
- Shiro
- Silk DI
- SLF4J
- Snippetory Template Engine
- SNMP4J
- Socket handler logging
- Spark
- Specsby
- Spock
- Spring
- Spring Batch
- Spring Boot
- Spring Boot Actuator
- Spring Boot Cache
- Spring Boot Flo
- Spring Cloud Config
- Spring Cloud Function
- Spring Data
- Spring Data JPA
- spring DI
- Spring Integration

- Spring JMX
- Spring Messaging Client
- Spring MVC
- Spring Properties
- Spring Scheduled
- Spring Security
- Spring Shell
- Spring Test
- Spring Transactions
- Spring Web
- SQLite Driver
- SSL
- Standard Widget Toolkit (SWT)
- Stateful (SFSB)
- Stateless (SLSB)
- Sticky Configured
- Stripes
- Struts
- SubCut
- Swagger
- SwarmCache
- Swing
- SwitchYard
- Syringe
- Talend ESB
- Teiid
- TensorFlow
- Test Interface
- TestNG

- Thymeleaf
- TieFaces
- tinylog
- Tomcat
- Tornado Inject
- Trimou
- Trunk JGuard
- Twirl
- Twitter Util Logging
- UberFire
- Unirest
- Unitils
- Vaadin
- Velocity
- Vlad
- Water Template Engine
- Web Services Metadata
- Web Session
- Web XML File
- WebLogic Web XML
- Webmacro
- WebSocket
- WebSphere EJB
- WebSphere EJB Ext
- WebSphere Web XML
- WebSphere WS Binding
- WebSphere WS Extension
- Weka
- Weld

- WF Core JTA
- Wicket
- Winter
- WSDL
- WSO2
- WSS4J
- XACML
- XFire
- XMLUnit
- Zbus Client
- Zipkin

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.
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.

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.

For more information on categorizing tasks, see [Using custom rule categories](#).

A.4. ADDITIONAL RESOURCES

A.4.1. Contributing to the project

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 must cover.
- Provide example applications to test migration rules.
- Identify application components and problem areas that might be difficult to migrate:
 - Write a short description of the problem migration areas.
 - Write a brief overview describing how to solve the problem in migration areas.
- Try Migration Toolkit for Runtimes on your application. Make sure to report any issues you meet.

- 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.

For more information, see [Rule 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. Migration Toolkit for Runtimes development resources

Use the following resources to learn and contribute to the Migration Toolkit for Runtimes development:

- 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 2024-06-12 18:33:08 UTC