Red Hat JBoss Migration Toolkit 3.0 Windup User Guide

Simplify Migration of Java Applications

Red Hat Customer Content
Services

Red Hat JBoss Migration Toolkit 3.0 Windup User Guide

Simplify Migration of Java Applications

Legal Notice

Abstract

This guide describes how to use Windup to simplify migration of Java applications.

Table of Contents

CHAPTER 1. INTRODUCTION

1.1. ABOUT THE WINDUP USER GUIDE

This guide is for engineers, consultants, and others who want to use Windup to migrate Java applications or other components. It describes how to install and run Windup, review the generated reports, and take advantage of additional features.

1.1.1. Use of WINDUP_HOME in This Guide

This guide uses the **`WINDUP_HOME`** replaceable value to denote the path to the Windup distribution. When you encounter this value in the guide, be sure to replace it with the actual path to your Windup installation.

- If you download and install the latest distribution of Windup, **`WINDUP_HOME`** refers to the **`windup-distribution-3.0.0.Final`** folder extracted from the downloaded ZIP file.
- If you build Windup from GitHub source, **`WINDUP_HOME`** refers to the **`windup-distribution-3.0.0.Final`** folder extracted from the **`windup-distribution/target/windup-distribution-3.0.0-SNAPSHOT-offline.zip`** file.

1.2. ABOUT WINDUP

What is Windup?

Windup is an extensible and customizable rule-based tool that helps simplify migration of Java applications.

Windup examines application artifacts, including project source directories and application archives, then produces an HTML report that highlights areas needing changes. Windup 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 Windup Simplify Migration?

Windup 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 JBoss EAP.

1.2.1. Windup Features

Windup provides a number of capabilities to assist with planning and executing migration projects.

Planning and work estimation

Windup assists project managers by detailing the type of work and estimation of effort to complete the tasks. Level of effort is represented in Windup reports as story points. Actual estimates will be based on the skills required and the classification of migration work needed.

Identifying migration issues and providing solutions

Windup identifies migration issues and highlights specific points in the code where an issue occurs. Windup suggests code changes and provides additional resources to help engineers resolve the specific issue.

Detailed reporting

Windup produces numerous reports to give both high-level views of the migration effort and details of specific migration tasks. You can view migration issues across all applications, charts and summary information about issues in an application, a breakdown of issues by module in the application, reports of technologies used, and dependencies on other applications and services. You can also examine source files to see the line of code where an issue occurs.

Built-in rules and migration paths

Windup comes with a core set of rules to provide migration assistance for several common migration paths. These rules identify the use of proprietary functionality from other application servers or deprecated subsystems from previous versions of JBoss EAP. Windup also contains rules to identify common migration issues, such as hard-coded IP addresses and JNDI lookups.

Rule extensibility and customization

Windup provides the ability to create powerful and complex rules. You can expand upon the core set of rules provided by Windup and create rules to identify additional issues that are important for your migration project. You can also override core rules and create custom rule categories. See the *Windup Rules Development Guide* for more information on customizing Windup rules.

Ability to analyze source code or application archives

Windup can evaluate application archives or source code, and can evaluate multiple applications together. It can identify archives that are shared across multiple applications, which can help with more accurate effort estimation.

1.2.2. Windup Rules

Windup is a rule-based migration tool that analyzes the APIs, technologies, and architectures used by the applications you plan to migrate. In fact, the Windup analysis process is implemented using Windup rules. Windup uses rules internally to extract files from archives, decompile files, scan and classify file types, analyze XML and other file content, analyze the application code, and build the reports.

Windup builds a data model based on the rule execution results and stores component data and relationships in a graph database, which can then be queried and updated as needed by the migration rules and for reporting purposes.

Windup rules use the following rule pattern:

```
when(condition)
  perform(action)
otherwise(action)
```

Windup provides a comprehensive set of standard migration rules out-of-the-box. Because applications may contain custom libraries or components, Windup allows you to write your own rules to identify use of components or software that may not be covered by the existing ruleset. If you plan to write your own custom rules, see the *Windup Rules Development Guide* for detailed instructions.

1.3. SUPPORTED CONFIGURATIONS

1.3.1. System Requirements

1.3.1.1. Software

- Java Platform, JRE version 8+
- Windup is tested on Linux, Mac OS X, and Windows. Other operating systems with Java 8+ support should work equally well.

1.3.1.2. Hardware

The following memory and disk space requirements are the minimum needed to run Windup. If your application is very large or you need to evaluate multiple applications, you may want to increase these values to improve performance. For tips on how to optimize performance, see Optimize Windup Performance.

- A minimum of 4 GB RAM. For better performance, a quad-core processor with 8 GB RAM is recommended. This allows 3 - 4 GB RAM for use by the JVM.
- A minimum of 4 GB of free disk space. A fast disk, especially a solid-state drive (SSD), should improve performance.

1.3.2. Supported Migration Paths

Windup supports application migration from several platforms to Red Hat JBoss Enterprise Application Platform (JBoss EAP). See the below table for which JBoss EAP version is currently supported by Windup for direct migration from your source platform.

Source Platform	Migration to JBoss EAP 6	Migration to JBoss EAP 7

Source Platform	Migration to JBoss EAP 6	Migration to JBoss EAP 7
Oracle® WebLogic Server	✔	✔
IBM® WebSphere® Application Server	✔	✔
JBoss EAP 4	✔	✘ [a]
JBoss EAP 5	✔	✔
JBoss EAP 6	N/A	✔

[a] Although Windup does not currently provide rules for this migration path, Red Hat Consulting can assist with migration from any source platform.

You will specify the source and target technologies by passing the `--source` and `--target` arguments to the Windup command.

Example Windup Command: WebLogic to JBoss EAP 6

```
$ WINDUP_HOME/bin/windup --source weblogic --target eap:6 --input
/path/to/application.war
```

Example Windup Command: JBoss EAP 6 to JBoss EAP 7

```
$ WINDUP_HOME/bin/windup --source eap:6 --target eap:7 --input
/path/to/application.war
```

See Run Windup for more information on running Windup.

CHAPTER 2. GETTING STARTED

2.1. INSTALL WINDUP

1. Download the latest Windup ZIP distribution.
2. Extract the ZIP file to a directory of your choice.

The extracted directory is known as **`WINDUP_HOME`** in this guide.

2.2. RUN WINDUP

Use the following steps to run Windup against your application.

1. Open a terminal and navigate to the **`WINDUP_HOME/bin/`** directory.
2. Execute the **`windup`** script, or **`windup.bat`** for Windows, and specify the appropriate arguments.

   ```
   $ ./windup --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. See the **`--input`** argument description.
   - **`--output`**: The output directory for the generated reports. See the **`--output`** argument description.
   - **`--source`**: The source technology for the application migration. See the **`--source`** argument description.
   - **`--target`**: The target technology for the application migration. See the **`--target`** argument description.
   - **`--packages`**: The packages to be evaluated. This argument is highly recommended to improve performance. See the **`--packages`** argument description.

   See Windup Command-line Arguments for a detailed description of all available command-line arguments.
3. Access the report.

See Windup Command Examples below for concrete examples of commands that use source code directories and archives located in the Windup GitHub repository.

Windup Command Examples

Running Windup on Source Code

The following command runs against the Windup seam-booking-5.2 test application source code. It evaluates all **`org.jboss.seam`** packages and creates a directory named 'seam-booking-report' in the **`/home/username/windup-reports/`** directory to contain the reporting output.

```
$ WINDUP_HOME/bin/windup --sourceMode --input /home/username/windup-
source/test-files/seam-booking-5.2/ --output /home/username/windup-
reports/seam-booking-report --target eap:6 --packages org.jboss.seam
```

Running Windup on an Application Archive

The following command runs against the Windup jee-example-app-1.0.0.ear test EAR archive. It evaluates all **`com.acme`** and **`org.apache`** packages and creates a directory named 'jee-example-app-1.0.0.ear-report' in the **`/home/username/windup-reports/`** directory to contain the reporting output.

```
$ WINDUP_HOME/bin/windup  --input /home/username/windup-source/test-
files/jee-example-app-1.0.0.ear --output /home/username/windup-
reports/jee-example-app-1.0.0.ear-report --target eap:6 --packages
com.acme org.apache
```

Override Windup 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:

```
$ WINDUP_HOME/bin/windup -Dwindup.decompiler=procyon --input
INPUT_ARCHIVE_OR_DIRECTORY --output OUTPUT_REPORT_DIRECTORY --target
TARGET_TECHNOLOGY --packages PACKAGE_1 PACKAGE_2
```

Windup Help

To see the complete list of available arguments for the **`windup`** command, open a terminal, navigate to the **`WINDUP_HOME`** directory, and execute the following command:

```
$ WINDUP_HOME/bin/windup --help
```

2.3. ACCESS THE REPORT

When you execute Windup, the report is generated in the **`OUTPUT_REPORT_DIRECTORY`** that you specify using the **`--output`** argument in the command line. Upon completion of execution, you will see the following message in the terminal with the location of the report.

```
Windup report created: OUTPUT_REPORT_DIRECTORY/index.html
              Access it at this URL:
file:///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
├── archives/           // Archives extracted from the application
├── mavenized/          // Optional Maven project structure
├── reports/            // Generated HTML reports
├── stats/              // Performance statistics
```

Use a browser to open the **`index.html`** file located in the report output directory. See Review the Reports for information on navigating the Windup reports.

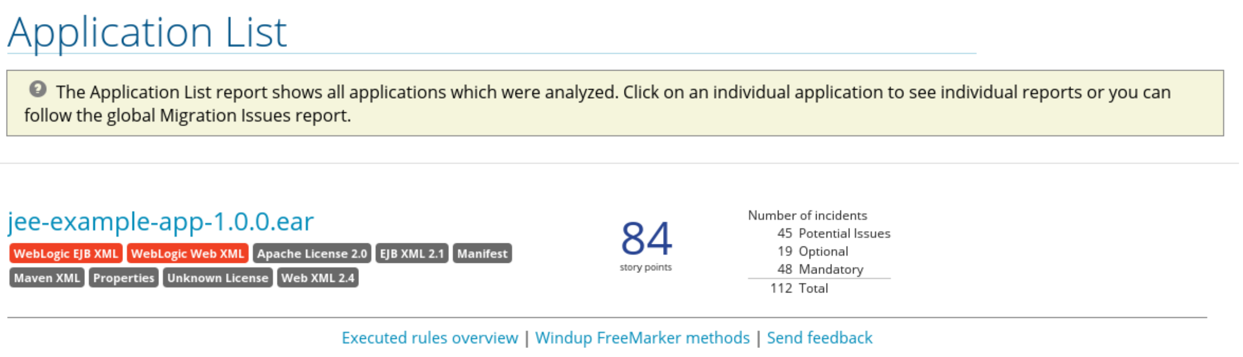
CHAPTER 3. REVIEW THE REPORTS

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 Windup GitHub source repository. The report was generated using the following command.

```
WINDUP_HOME/bin/windup --input /home/username/windup-source/test-
files/jee-example-app-1.0.0.ear/ --output /home/username/windup-
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



Application List

The Application List report shows all applications which were analyzed. Click on an individual application to see individual reports or you can follow the global Migration Issues report.

jee-example-app-1.0.0.ear

WebLogic EJB XML | WebLogic Web XML | Apache License 2.0 | EJB XML 2.1 | Manifest | Maven XML | Properties | Unknown License | Web XML 2.4

84 story points

```
Number of incidents
     45  Potential Issues
     19  Optional
     48  Mandatory
    112  Total
```

Executed rules overview | Windup FreeMarker methods | Send feedback

Note

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

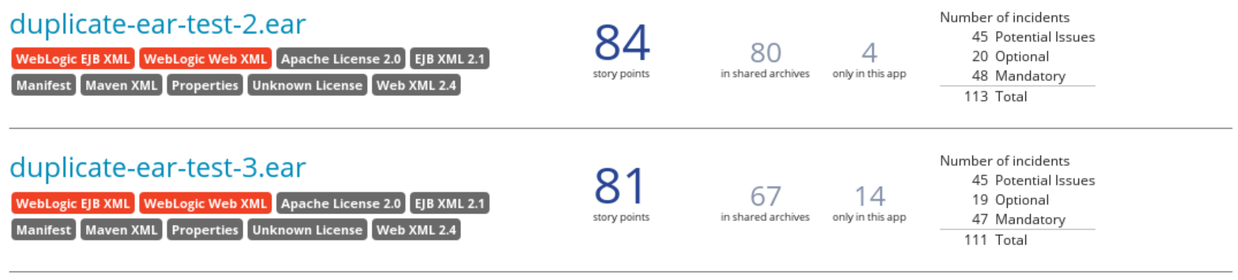
Click on the name of the application, **jee-example-app-1.0.0.ear**, to view the application report. The following table lists all of the reports that can be access from this main Windup landing page.

Report	How to Access the Report
Application	Click on the name of the application.
Archives shared by multiple applications	Click on the **Archives shared by multiple applications** link. Note that this link is only available when there are shared archives across multiple applications.
Rule Provider Executions	Click on the **Executed rules overview** link at the bottom of the page.

Report	How to Access the Report
Windup Freemarker Functions and Directives	Click on the **Windup FreeMarker methods** link at the bottom of the page.
Send Feedback Form	Click on the **Send feedback** link at the bottom of the page to open a form that allows you to submit feedback to the Windup team.

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



duplicate-ear-test-2.ear

WebLogic EJB XML | WebLogic Web XML | Apache License 2.0 | EJB XML 2.1 | Manifest | Maven XML | Properties | Unknown License | Web XML 2.4

84 story points | 80 in shared archives | 4 only in this app

Number of incidents
45 Potential Issues
20 Optional
48 Mandatory
113 Total

duplicate-ear-test-3.ear

WebLogic EJB XML | WebLogic Web XML | Apache License 2.0 | EJB XML 2.1 | Manifest | Maven XML | Properties | Unknown License | Web XML 2.4

81 story points | 67 in shared archives | 14 only in this app

Number of incidents
45 Potential Issues
19 Optional
47 Mandatory
111 Total

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

3.1. APPLICATION REPORT

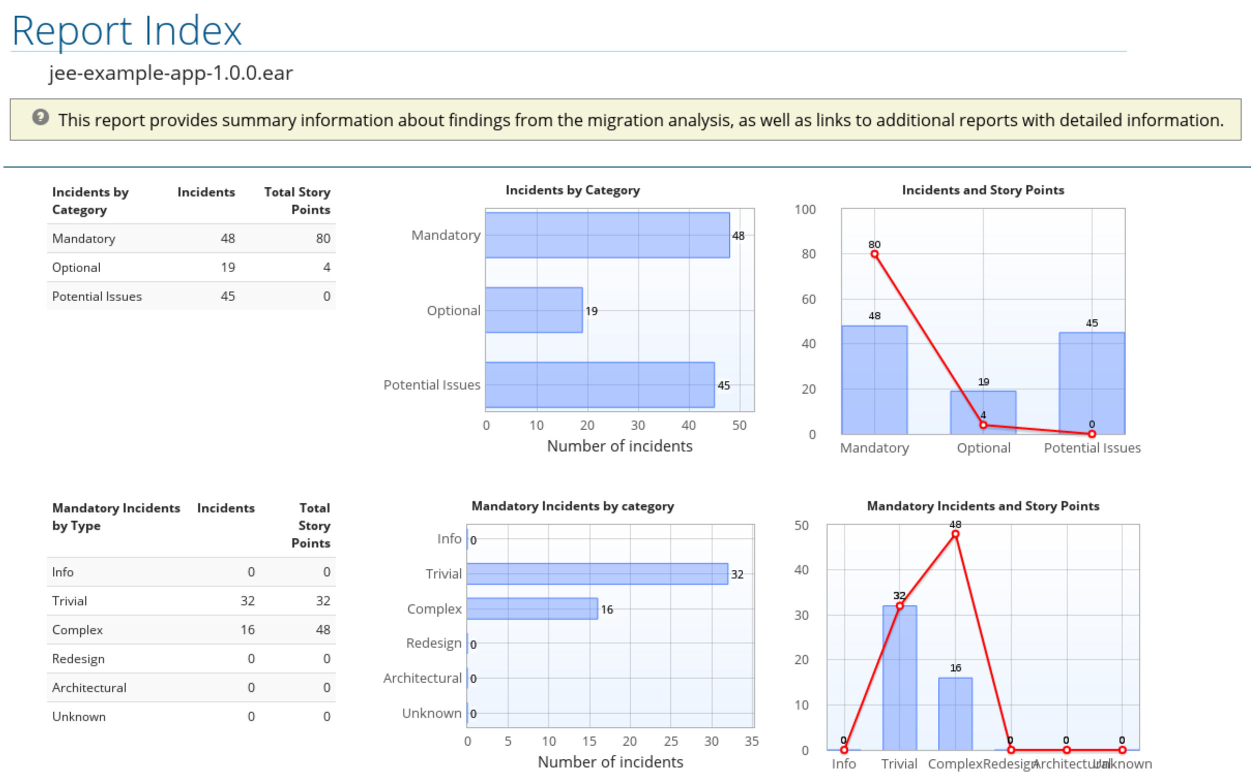
3.1.1. Report Index

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

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

- The incidents and story points by category (Mandatory, Optional, or Potential Issues)
- The incidents and story points by level of effort of the suggested changes
- The incidents by package

Figure 3.3. Report Index



Report Index

jee-example-app-1.0.0.ear

This report provides summary information about findings from the migration analysis, as well as links to additional reports with detailed information.

Incidents by Category	Incidents	Total Story Points
Mandatory	48	80
Optional	19	4
Potential Issues	45	0

Mandatory Incidents by Type	Incidents	Total Story Points
Info	0	0
Trivial	32	32
Complex	16	48
Redesign	0	0
Architectural	0	0
Unknown	0	0

Incidents by Category
Number of incidents
Incidents and Story Points
Mandatory Incidents by category
Number of incidents
Mandatory Incidents and Story Points

At the bottom of the page is a list of 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
Migration 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.
Unparsable	Shows all files that Windup 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.
Dependencies	Displays all Java-packaged dependencies found within the application.
Remote Services	Displays all remote services references that were found within the application.
EJBs	Contains a list of EJBs found within the application.

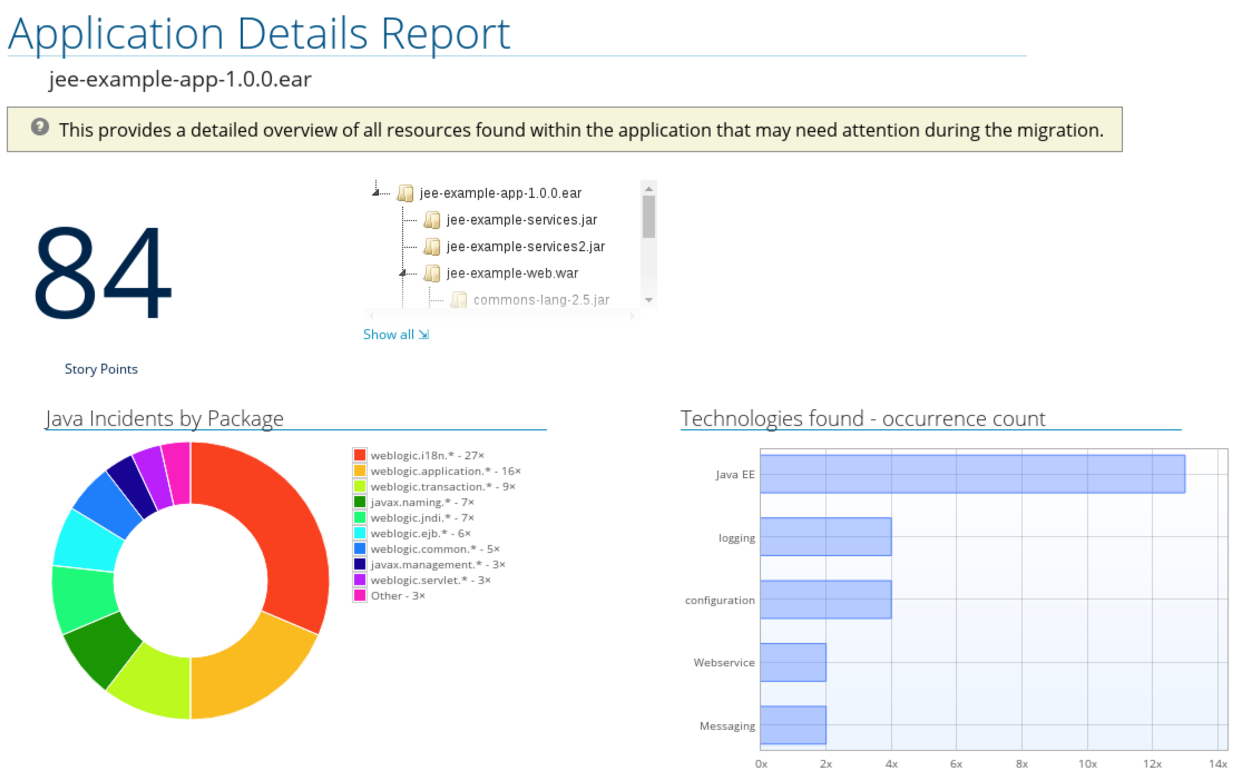
Report	Description
JBPM	Contains all of the JBPM-related resources that were discovered during analysis.
JPA	Contains details on all JPA-related resources that were found in the application.
Hibernate	Contains details on all Hibernate-related resources that were found in the application.
Server Resources	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 Windup configuration, were not processed. See the `--userIgnorePath` option for more information.
About	Describes the current version of Windup and provides helpful links for further assistance.

3.1.2. Application Details Report

Access this report from the report index 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 3.4. Application Details Report



Application Details Report

jee-example-app-1.0.0.ear

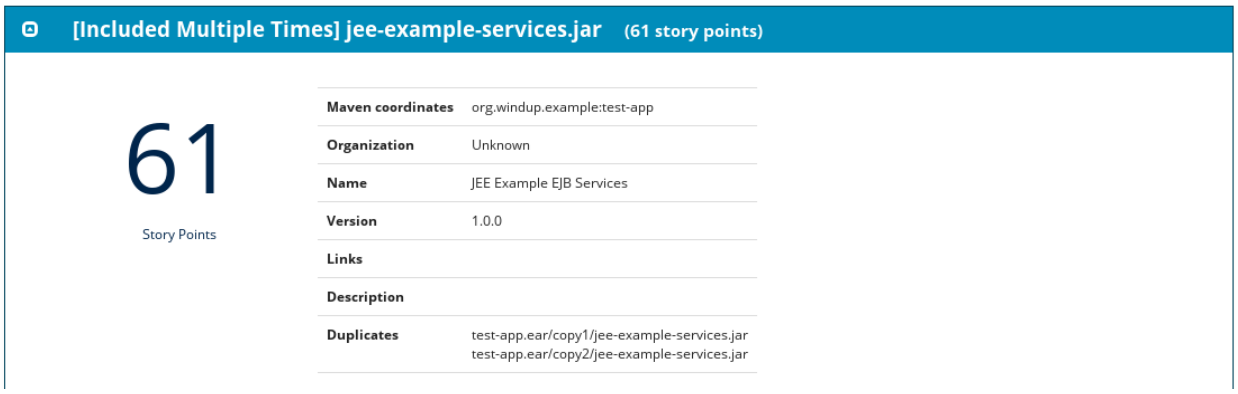
This provides a detailed overview of all resources found within the application that may need attention during the migration.

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: Java Source, Decompiled Java File, Manifest, Properties, EJB XML, Spring XML, Web XML, Hibernate Cfg, Hibernate Mapping
Issues	Warnings about areas of code that need review or changes.
Story Points	Level of effort required to migrate the file. See Rule Story Points for more details.

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 3.5. Duplicate Archive in an Application



[Included Multiple Times] jee-example-services.jar (61 story points)

61

Story Points

Maven coordinates	org.windup.example:test-app
Organization	Unknown
Name	JEE Example EJB Services
Version	1.0.0
Links	
Description	
Duplicates	test-app.ear/copy1/jee-example-services.jar test-app.ear/copy2/jee-example-services.jar

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.

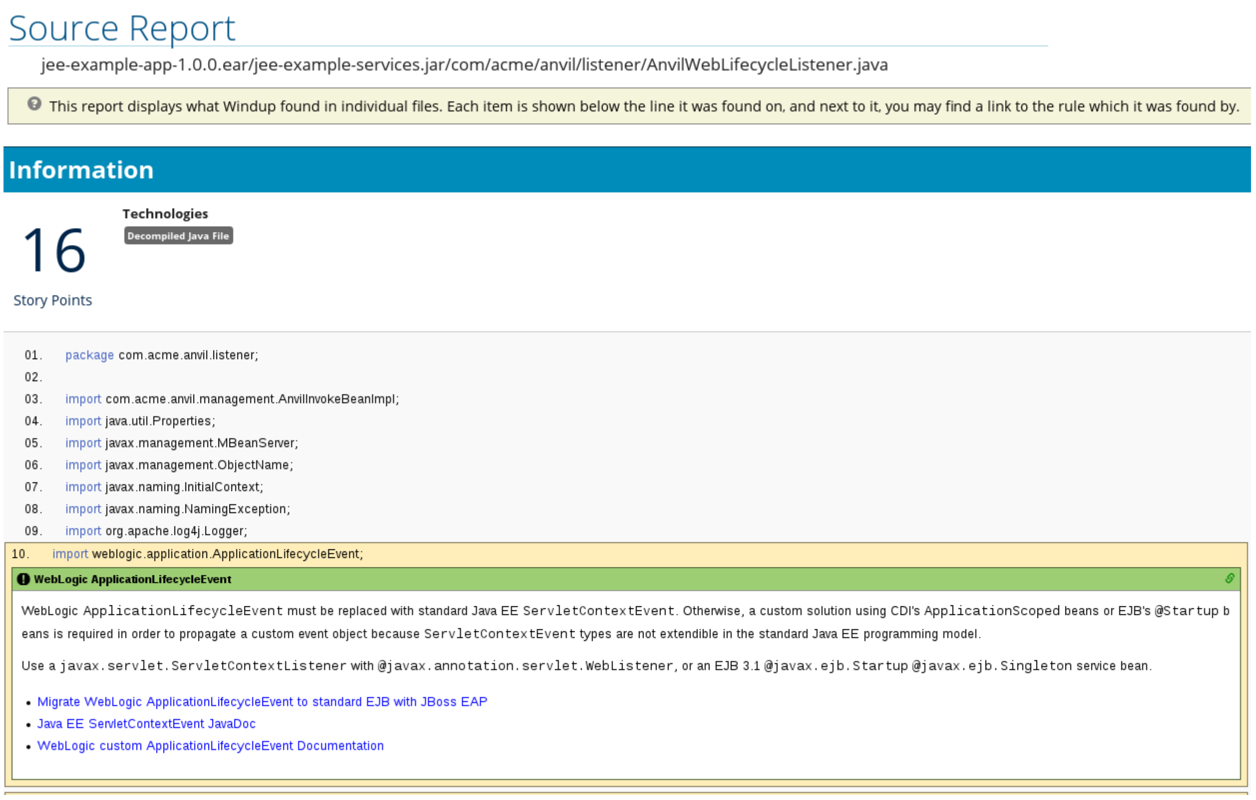
3.1.3. Source Report

The analysis of the `jee-example-services.jar` lists the files in the JAR and the warnings and story points assigned to each one. Notice the `com.acme.anvil.listener.AnvilWebLifecycleListener` file, at the time of this test, has 22 warnings and is assigned 16 story points. Click on the file link to see the detail.

- The **Information** section provides a summary of the story points and notes that the file was decompiled by Windup.
- This is followed by the file source code listing. Warnings appear in the file at the point where migration is required.

In this example, warnings appear at various import statements, declarations, and method calls. Each warning describes the issue and the action that should be taken.

Figure 3.6. Source Report



Source Report

jee-example-app-1.0.0.ear/jee-example-services.jar/com/acme/anvil/listener/AnvilWebLifecycleListener.java

This report displays what Windup found in individual files. Each item is shown below the line it was found on, and next to it, you may find a link to the rule which it was found by.

Information

16

Story Points

Technologies

Decompiled Java File

```
package com.acme.anvil.listener;

import com.acme.anvil.management.AnvilInvokeBeanImpl;
import java.util.Properties;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import org.apache.log4j.Logger;
import weblogic.application.ApplicationLifecycleEvent;
```

WebLogic ApplicationLifecycleEvent

WebLogic `ApplicationLifecycleEvent` must be replaced with standard Java EE `ServletContextEvent`. Otherwise, a custom solution using CDI's `ApplicationScoped` beans or EJB's `@Startup` beans is required in order to propagate a custom event object because `ServletContextEvent` types are not extendible in the standard Java EE programming model.

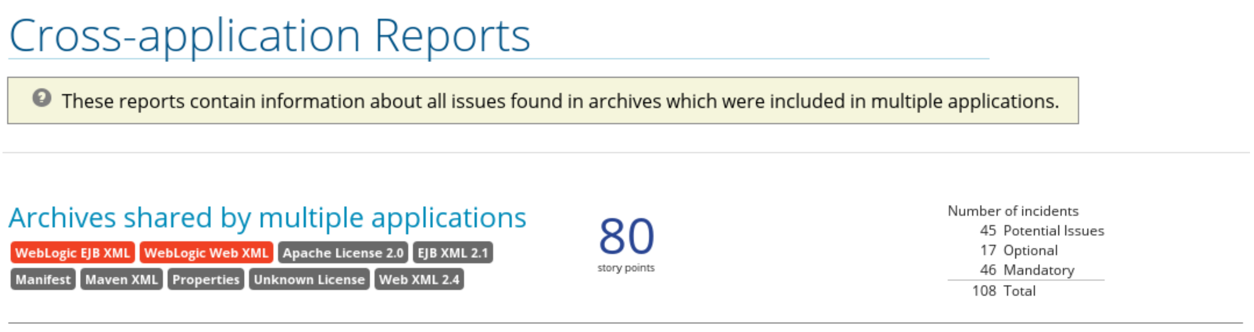
Use a `javax.servlet.ServletContextListener` with `@javax.annotation.servlet.WebListener`, or an EJB 3.1 `@javax.ejb.Startup` `@javax.ejb.Singleton` service bean.

- Migrate WebLogic ApplicationLifecycleEvent to standard EJB with JBoss EAP
- Java EE ServletContextEvent JavaDoc
- WebLogic custom ApplicationLifecycleEvent Documentation

3.2. 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 3.7. Archives Shared by Multiple Applications



Cross-application Reports

These reports contain information about all issues found in archives which were included in multiple applications.

Archives shared by multiple applications

WebLogic EJB XML | WebLogic Web XML | Apache License 2.0 | EJB XML 2.1 | Manifest | Maven XML | Properties | Unknown License | Web XML 2.4

80 story points

Number of incidents	
45	Potential Issues
17	Optional
46	Mandatory
108	Total

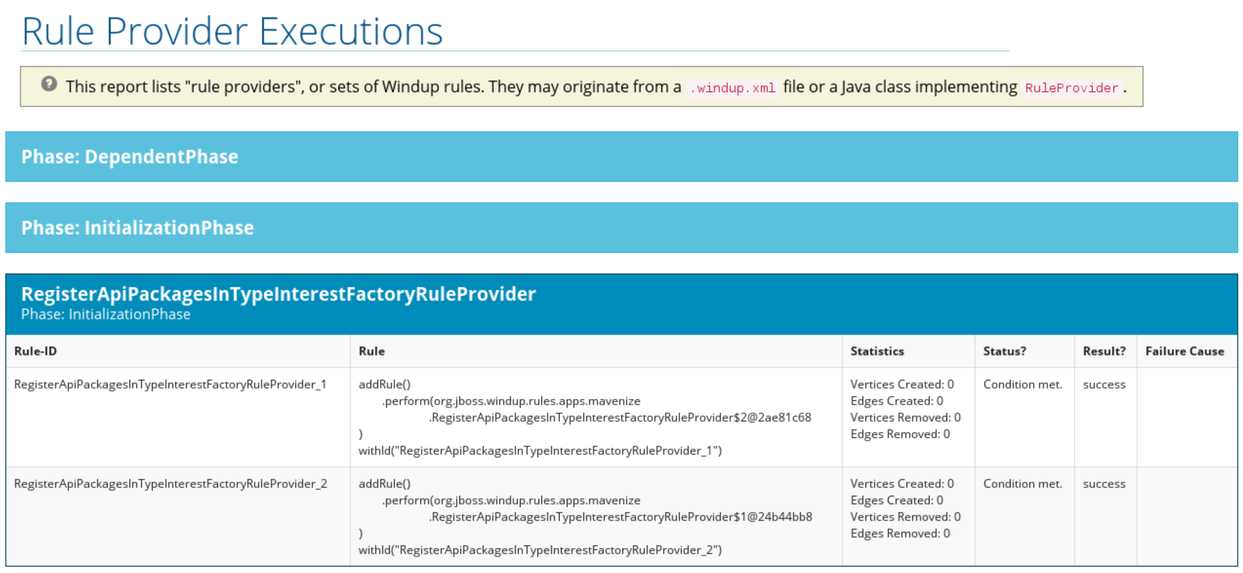
This allows you to view the detailed reports for all archives that are shared across multiple applications.

3.3. RULE PROVIDER EXECUTION REPORT

Access this report from the report landing page by clicking the **Executed rules overview** link.

This report provides the list of rules that executed when running the Windup migration command against the application.

Figure 3.8. Rule Provider Report



Rule Provider Executions

This report lists "rule providers", or sets of Windup rules. They may originate from a `.windup.xml` file or a Java class implementing `RuleProvider`.

Phase: DependentPhase

Phase: InitializationPhase

RegisterApiPackagesInTypeInterestFactoryRuleProvider
Phase: InitializationPhase

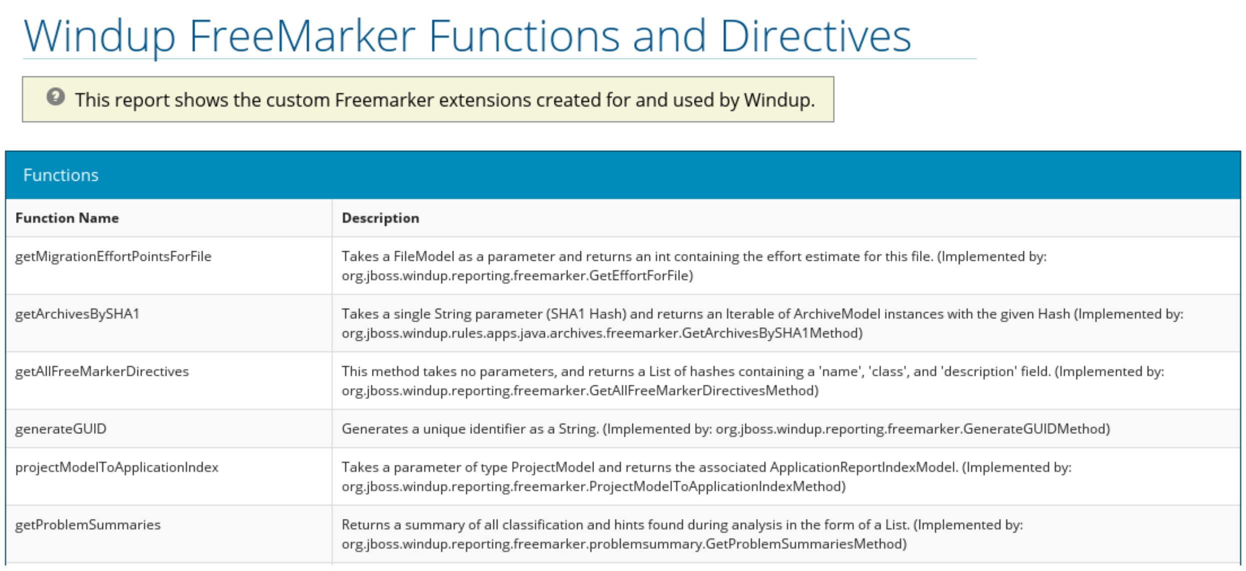
Rule-ID	Rule	Statistics	Status?	Result?	Failure Cause
RegisterApiPackagesInTypeInterestFactoryRuleProvider_1	addRule() .perform(org.jboss.windup.rules.apps.mavenize .RegisterApiPackagesInTypeInterestFactoryRuleProvider$2@2ae81c68) withId("RegisterApiPackagesInTypeInterestFactoryRuleProvider_1")	Vertices Created: 0 Edges Created: 0 Vertices Removed: 0 Edges Removed: 0	Condition met.	success	
RegisterApiPackagesInTypeInterestFactoryRuleProvider_2	addRule() .perform(org.jboss.windup.rules.apps.mavenize .RegisterApiPackagesInTypeInterestFactoryRuleProvider$1@24b44bb8) withId("RegisterApiPackagesInTypeInterestFactoryRuleProvider_2")	Vertices Created: 0 Edges Created: 0 Vertices Removed: 0 Edges Removed: 0	Condition met.	success	

3.4. WINDUP FREEMARKER FUNCTIONS AND DIRECTIVES REPORT

Access this report from the report landing page by clicking the **Windup FreeMarker methods** link.

This report lists all the registered functions and directives that were used to build the report. It is useful if you plan to build your own custom report or for debugging purposes.

Figure 3.9. FreeMarker Functions and Directives


Windup FreeMarker Functions and Directives

This report shows the custom Freemarker extensions created for and used by Windup.

Functions

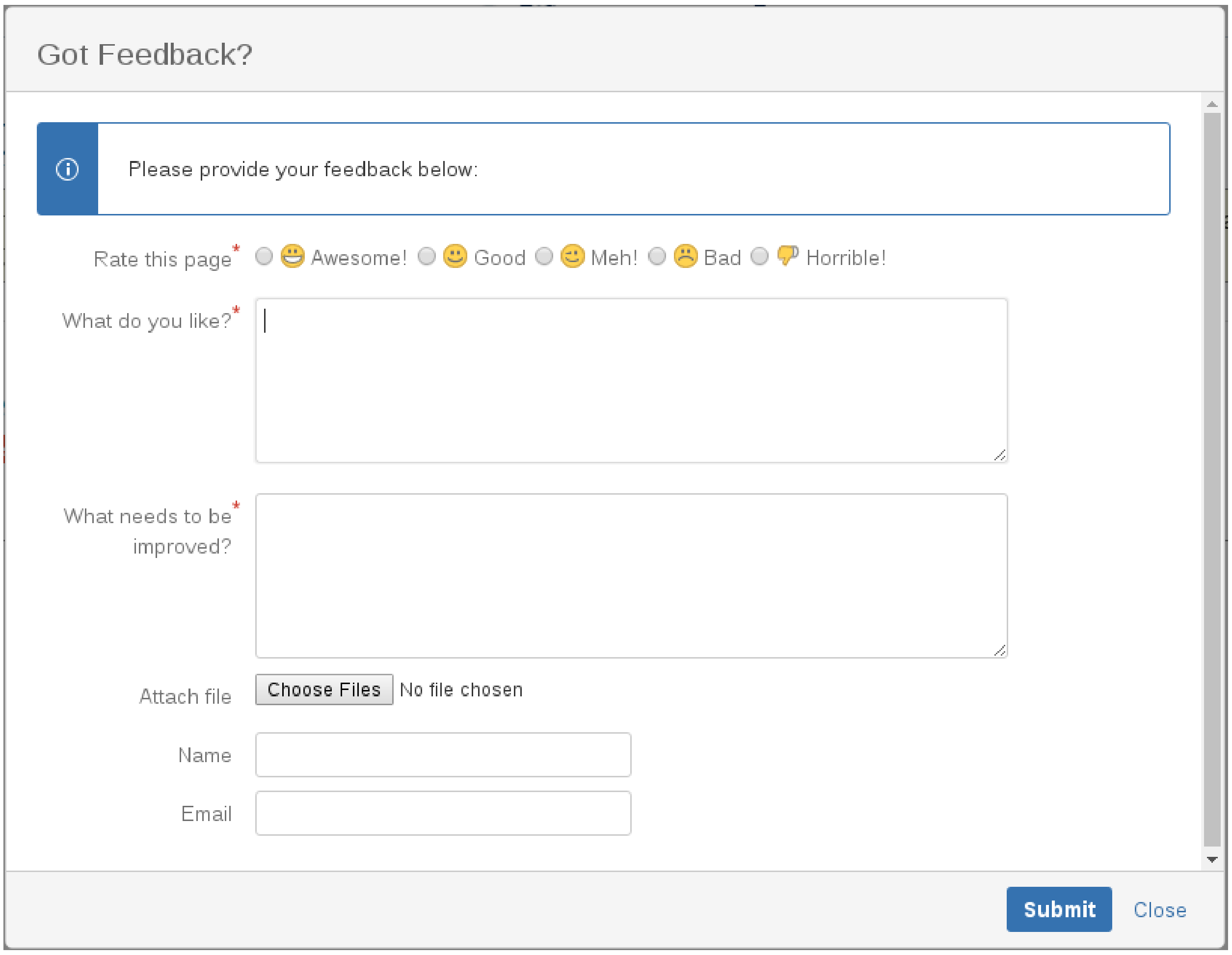
Function Name	Description
getMigrationEffortPointsForFile	Takes a FileModel as a parameter and returns an int containing the effort estimate for this file. (Implemented by: org.jboss.windup.reporting.freemarker.GetEffortForFile)
getArchivesBySHA1	Takes a single String parameter (SHA1 Hash) and returns an Iterable of ArchiveModel instances with the given Hash (Implemented by: org.jboss.windup.rules.apps.java.archives.freemarker.GetArchivesBySHA1Method)
getAllFreeMarkerDirectives	This method takes no parameters, and returns a List of hashes containing a 'name', 'class', and 'description' field. (Implemented by: org.jboss.windup.reporting.freemarker.GetAllFreeMarkerDirectivesMethod)
generateGUID	Generates a unique identifier as a String. (Implemented by: org.jboss.windup.reporting.freemarker.GenerateGUIDMethod)
projectModelToApplicationIndex	Takes a parameter of type ProjectModel and returns the associated ApplicationReportIndexModel. (Implemented by: org.jboss.windup.reporting.freemarker.ProjectModelToApplicationIndexMethod)
getProblemSummaries	Returns a summary of all classification and hints found during analysis in the form of a List. (Implemented by: org.jboss.windup.reporting.freemarker.problemsummary.GetProblemSummariesMethod)

3.5. SEND FEEDBACK FORM

Access this feedback form from the report landing page by clicking the **Send feedback** link.

This form allows you to rate the product, talk about what you like and suggestions for improvements.

Figure 3.10. Send Feedback Form



Got Feedback?

Please provide your feedback below:

Rate this page* ○ Awesome! ○ Good ○ Meh! ○ Bad ○ Horrible!

What do you like?*

What needs to be improved?*

Attach file: Choose Files No file chosen

Name

Email

CHAPTER 4. EXPORT THE REPORT IN CSV FORMAT

Windup 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 a Windup report.

4.1. EXPORT THE REPORT

To export the report into a CSV file, run Windup with **`--exportCSV`** argument. The CSV file will be created in the directory specified by the **`--output`** argument.

4.2. 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.

4.3. 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.

CHAPTER 5. MAVENIZE YOUR APPLICATION

Windup 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. GENERATE THE MAVEN PROJECT STRUCTURE

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

The following example runs Windup using the jee-example-app-1.0.0.ear test application.

```
$ WINDUP_HOME/bin/windup --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 Windup against source code.

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

5.2. REVIEW THE MAVEN PROJECT STRUCTURE

The **`/path/to/output/mavenized/APPLICATION_NAME/`** directory will contain 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. OPTIMIZE WINDUP PERFORMANCE

Windup 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, Windup 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 Windup.

6.1. TIPS TO OPTIMIZE PERFORMANCE

6.1.1. Application and Command-line Suggestions

Try these suggestions first before upgrading hardware.

- If possible, execute Windup 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 Windup using the **`--packages`** argument on the **`WINDUP_HOME/bin/windup`** command line. If you omit this argument, Windup will decompile everything, which has a big impact on performance.
- Specify the **`--excludePackages`** and **`--excludeTags`** arguments where possible to exclude them from processing.
- Add additional proprietary packages that should not be processed to the **`ignore/proprietary.package-ignore.txt`** file in the Windup distribution directory. Windup can still find the references to the packages in the application source code, but avoids the need to decompile and analyze the proprietary classes.
- Increase your ulimit when analyzing large applications.
- If you have access to a server that has better resources than your laptop or desktop machine, you may want to consider running Windup on that server.

6.1.2. Hardware Upgrade Suggestions

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 Windup 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.

APPENDIX A. REFERENCE MATERIAL

A.1. WINDUP COMMAND-LINE ARGUMENTS

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

Note

To run the Windup command without prompting, for example when executing from a script, use **`--batchMode`** to take the default values for unspecified parameters and **`--overwrite`** to force delete the output directory. Also be sure to specify the required**`--input`** and **`--target`** arguments.

See the description for each argument for more details.

Table A.1. Windup CLI Arguments

Argument	Description
--additionalClassPath	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.
--addonDir	Add the specified directory as a custom add-on repository.
--batchMode	Flag to specify that Windup 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.
--debug	Flag to run Windup in debug mode.
--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.

Argument	Description
--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 generate a Tattletale report for each application.
--excludePackages	A space-delimited 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 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. See the Input File Argument Tables for details.
--exportCSV	Flag to export the report data to a CSV file on your local file system. Windup creates the file in the directory specified by the **`--output`** 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 .
--help	Display the Windup help message.
--immutableAddonDir	Add the specified directory as a custom read-only add-on repository.
--includeTags	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 **`--listTags`** argument.

Argument	Description
--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. See Specify the Input for more information.
--install	Specify add-ons to install. The syntax is **`GROUP_ID:ARTIFACT_ID[:VERSION]`**. For example, **`--install core-addon-x`** or **`--install org.example.addon:example:1.0.0`**.
--keepWorkDirs	Flag to instruct Windup to not delete temporary working files, such as the graph database and unzipped archives. This is useful for debugging purposes.
--list	Flag to list installed add-ons.
--listSourceTechnologies	Flag to list all available source technologies.
--listTags	Flag to list all available tags.
--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 **`--mavenizeGroupId`** option.
--mavenizeGroupId	When used with the **`--mavenize`** option, all generated **`pom.xml`** files will use the provided value for their **`<groupId>`**. If this argument is omitted, Windup will attempt to determine an appropriate **`<groupId>`** based on the application, or will default to **`com.mycompany.mavenized`**.

Argument	Description
--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 Windup. See Specify the Output Directory for more information.
--overwrite	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. **Warning** Be careful not to specify a report output directory that contains important information!
--packages	A space-delimited list of the packages to be evaluated by Windup. It is highly recommended to use this argument. See Select Packages for more information.
--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
--source	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. 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. See the Input File Argument Tables for details.
--target	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. See Set the Target Technology for more information.
--userIgnorePath	Specify a location, in addition to **`${user.home}/.windup/ignore/`**, for Windup to identify files that should be ignored.
--userRulesDirectory	Specify a location, in addition to **`WINDUP_HOME/ignore/`** and **`${user.home}/.windup/rules/`**, for Windup to look for custom Windup rules. The value can be a directory containing ruleset files or a single ruleset file. The ruleset files must end in **`.windup.xml`**.
--version	Display the Windup version.

A.1.1. Specify 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. Specify the Output Directory

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

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, Windup will proceed to delete and recreate the directory. See the description of this argument for more information.

A.1.3. Set 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 Windup 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 for which JBoss EAP version is appropriate for your source platform.

A.1.4. Set 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 Windup 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 for which JBoss EAP version is appropriate for your source platform.

A.1.5. Select Packages

A space-delimited list of the packages to be evaluated by Windup. 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. It is best to provide this argument with one or more packages. For additional tips on how to improve performance, see Optimize Windup Performance.

A.2. RULE STORY POINTS

A.2.1. 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.

Windup 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.2.2. 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 Windup 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.2.3. Task Severity

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 indicate whether a task must be completed or can be postponed.

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

For more information on categorizing tasks, see Using Custom Rule Categories in the *Windup Rules Development Guide*.

A.3. ADDITIONAL RESOURCES

A.3.1. Get Involved

To help make Windup cover most application constructs and server configurations, including yours, you can help with any of the following items.

- Send an email to windup-users@lists.jboss.org and let us know what Windup 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 Windup on your application. Be sure to report any issues you encounter.
- Contribute to the Windup rules repository.

  - Write a Windup rule to identify or automate a migration process.
  - Create a test for the new rule.
  - Details are provided in the *Windup Rules Development Guide*.
- Contribute to the project source code.
  - Create a core rule.
  - Improve Windup performance or efficiency.
  - See the *Windup Core Development Guide* for information about how to configure your environment and set up the project.

Any level of involvement is greatly appreciated!

A.3.2. Important Links

- Windup wiki: https://github.com/windup/windup/wiki
- Windup forums: https://community.jboss.org/en/windup
- Windup JIRA issue trackers
  - Core Windup: https://issues.jboss.org/browse/WINDUP
  - Windup Rules: https://issues.jboss.org/browse/WINDUPRULE
- Windup users mailing List: windup-users@lists.jboss.org
- Windup on Twitter: @JBossWindup
- Windup IRC channel: Server FreeNode (**`irc.freenode.net`**), channel **`#windup`** (transcripts).

A.3.3. Known Windup Issues

You can review known issues for Windup here: Open Windup issues.

A.3.4. Report Issues with Windup

Windup uses JIRA as its issue tracking system. If you encounter an issue executing Windup, 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.3.4.1. 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 Windup issues, choose *Windup: (WINDUP)*.

     For issues with Windup rules, choose: *Windup 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 Windup development team, attach it to the issue using the **browse** button.

4. Click the **Create** button to create the JIRA issue.

Revised on 2017-02-17 00:07:40 EST