Windup User Guide

Introduction

This guide is for engineers, consultants, and others who plan to use Windup to migrate Java applications or other components.

What is Windup?

Overview

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

Running from a Forge environment, Windup examines application artifacts, including project source directories and applications archives, then produces an HTML report highlighting areas that need 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.

Features of Windup

Shared Data Model

Windup creates a shared data model graph that provides the following benefits.

- It enables complex rule interaction, allowing rules to pass findings to other rules.
- It enables 3rd-party plug-ins to interact with other plug-ins, rules and reports.
- The findings in data graph model can be searched and queried during rule execution and used for reporting purposes.

Extensibility

Windup can be extended by developers, users, and 3rd-party software.

- It provides a plug-in API to inject other applications into Windup.
- It enables 3rd-parties to create simple POJO plug-ins that can interact with the data graph.
- Means we don't have to invent everything. Users with domain knowledge can implement their own rules.

Better Rules

Windup provides more powerful and complex rules.

- XML-based rules are simple to write and and easy to implement.
- Java-based rule add-ons are based on OCPsoft Rewrite and provide greater flexibility and power creating when rules.
- Rules can now be nested to handle more complex situations. This means you can nest simple statements rather than use complex XPATH or REGEX expressions. *Rules can be linked using and/or statements

Automation

Windup has the ability to automate some of the migration processes.

- Windup is integrated with Forge 2, meaning it can generate projects, libraries,

and configuration files.

- Rules can create Forge inputs and put them into the data graph.
- During the automation phase, the data graph inputs can be processed to generate a new project.

Work Estimation

Estimates for the *level of effort* are based on the skills required and the classification of migration work needed. *Level of effort* is represented as *story points* in the Windup reports.

Better Reporting

Windup reports are now targeted for specific audiences.

- IT Management - Applications are ranked by cost of migration.
- Project Management - Reports detail the type of work and estimation of effort to complete the tasks.
- Developers - An Eclipse plug-in provides hints and suggested code changes within the IDE.

About 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 tool executes its own core set of rules through all phases of the migration process. It uses rules 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 familiar rule pattern:

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

```
otherwise(action)
```

Windup provides 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.

System Requirements

Software

- Java Platform, Enterprise Edition 7
- Windup is tested on Linux, Mac OS X, and Windows. Other Operating Systems with Java 7 support should work equally well.

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 4-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), will improve performance.

About the WINDUP_HOME Variable

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

- If you download and install the latest distribution of Windup from the JBoss Nexus repository, WINDUP_HOME refers to the windup-distribution-2.3.0-Final folder extracted from the downloaded ZIP file.
- If you build Windup from GitHub source, WINDUP_HOME refers to the windup-

distribution-2.3.0-Final folder extracted from the windup-distribution/target/windup-distribution-2.3.0-Final.zip file.

Get Started

Install Windup

1. If you installed previous versions of Windup, delete the `${user.home}/.windup/` directory. Otherwise you may see errors when you execute Windup.
2. Download the latest Windup ZIP distribution.
3. Extract the ZIP file in to a directory of your choice.

Execute Windup

Overview

These instructions use the replaceable variable `WINDUP_HOME` to refer to the fully qualified path to your Windup installation. For more information, see About the WINDUP_HOME Variable.

Run Windup

1. Open a terminal and navigate to the WINDUP_HOME directory.
2. The command to run Windup uses the following syntax.

```
For Linux:     $ bin/windup --input INPUT_ARCHIVE --output OUTPUT_REPORT --packages
For Windows:   > bin\windup.bat --input INPUT_ARCHIVE --output OUTPUT_REPORT --packa
```

3. This command takes arbitrary options processed by different add-ons. The list of options in the core Windup distribution can be found in the Javadoc. Most commonly used command line arguments are:

   --input INPUT_ARCHIVE_OR_FOLDER

   This is the fully qualified path of the application archive or folder you plan to migrate.

--output OUTPUT_REPORT_DIRECTORY (optional)

This is the fully qualified path to the folder that will contain the the report information produced by Windup.

- If omitted, the report will be generated in a **INPUT_ARCHIVE_OR_FOLDER.report** folder.
- If the output directory exists, you will see the following error.

```
***ERROR*** Files exist in /home/username/OUTPUT_REPORT_DIRECTORY, but --
overwrite not specified. Aborting!
```

You must specify the `--overwrite` argument to proceed. This forces Windup to delete and recreate the folder.

> WARNING
>
> Be careful not to specify a report output directory that contains important information!

--source (optional)

One or more source technologies, servers, platforms, or frameworks to migrate from.

> TIP
>
> For the list of the available `--source` servers or frameworks, use the `--listSourceTechnologies` argument on the `windup` command line as in the following example.
>
> ```
> bin/windup --listSourceTechnologies
> ```

--target (optional)

One or more source technologies, servers, platforms, or frameworks to migrate to.

TIP

For the list of the available `--target` servers or frameworks, use the `--listTargetTechnologies` argument on the `windup` command line as in the following example.

```
bin/windup --listTargetTechnologies
```

--overwrite (optional)

Specify this optional argument only if you are certain you want to force Windup to delete the existing **OUTPUT_REPORT_DIRECTORY** folder. The default value is `false`.

--userRulesDirectory (optional)

The fully qualified path to a user directory containing custom XML rules that should be loaded and executed by Windup. The XML ruleset files must use one of the following extensions: `*.windup.groovy` or `*.windup.xml`.

--packages PACKAGE_1, PACKAGE_2, PACKAGE_N (optional)

This is a comma-delimited list of the packages to be evaluated by Windup.

- In most cases, you are interested only in evaluating the custom application class packages and not the standard Java EE or 3rd party packages. For example, if the *MyCustomApp* application uses the package `com.mycustomapp`, you provide that package using the `--packages` argument on the command line.
- It is not necessary to provide the standard Java EE packages, like `java.util` or `javax.ejb`.
- While you can provide package names for standard Java EE 3rd 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, resulting in very slow performance. It is best to provide the argument with one or more packages.

--excludePackages PACKAGE_1, PACKAGE_2, PACKAGE_N (optional)

This is a comma-delimited list of the packages to be excluded by Windup.

--sourceMode (optional)

This argument is optional and is only required if the application to be evaluated contains source files rather than compiled binaries. The default value is `false`.

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

```
bin/windup -Dwindup.decompiler=procyon INPUT_ARCHIVE_OR_FOLDER --output OUTPUT_REPOR
```

5. To evaluate an application archive, use the following syntax:

```
bin/windup --input INPUT_ARCHIVE_OR_FOLDER --output OUTPUT_REPORT_DIRECTORY --packag
```

   To run Windup against application source code, you must add the `--sourceMode` argument:

```
bin/windup  --sourceMode --source SOURCE_TECHNOLOGY --target TARGET_TECHNOLOGY --inp
```

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

6. You should see the following result upon completion of the command:

```
**SUCCESS*** Windup report created: PATH_TO_REPORTS/index.html
             Access it at this URL: file:///home/username/PATH_TO_REPORTS/index.htm
```

WARNING	Depending on the size of the application and the hardware Windup is running on, this command can take a very long time. For tips on how to improve performance, see Optimize Windup Performance.

7. Open the `OUTPUT_REPORT_DIRECTORY/index.html` file in a browser to access the report. The following subdirectories in the `OUTPUT_REPORT_DIRECTORY` contain the supporting information for the report:

```
OUTPUT_REPORT_DIRECTORY/
    graph/
    renderedGraph/
    reports/
    stats/
    index.html
```

8. For details on how to evaluate the report data, seeReview the Report.

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:

```
bin/windup --help
```

Windup Command Examples

The following examples report against applications located in the Windup source test-files directory.

Source Code Example

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

```
bin/windup --sourceMode --input /home/username/windup-source/test-files/seam-booking-
5.2/ --output /home/username/windup-reports/seam-booking-report --source eap4,eap5 --
```

```
target eap6 --packages org.jboss.seam
```

Archive Example

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

```
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 --
source eap4,eap5 --target eap6 --packages com.acme org.apache
```

Windup Quickstart Examples

For more concrete examples, see the Windup quickstarts located on GitHub here: https://github.com/windup/windup-quickstarts. If you prefer, you can download the latest release ZIP or TAR distribution of the quickstarts.

The quickstarts provide examples of Java-based and XML-based rules you can run and test using Windup. The README instructions provide a step-by-step guide to run the quickstart example. You can also look through the code examples and use them as a starting point for creating your own rules.

Review the Report

About the Report

When you execute Windup, the report is generated in the `OUTPUT_REPORT_DIRECTORY` you specify for the `--output` argument in the command line. This output directory contains the following files and subdirectories:

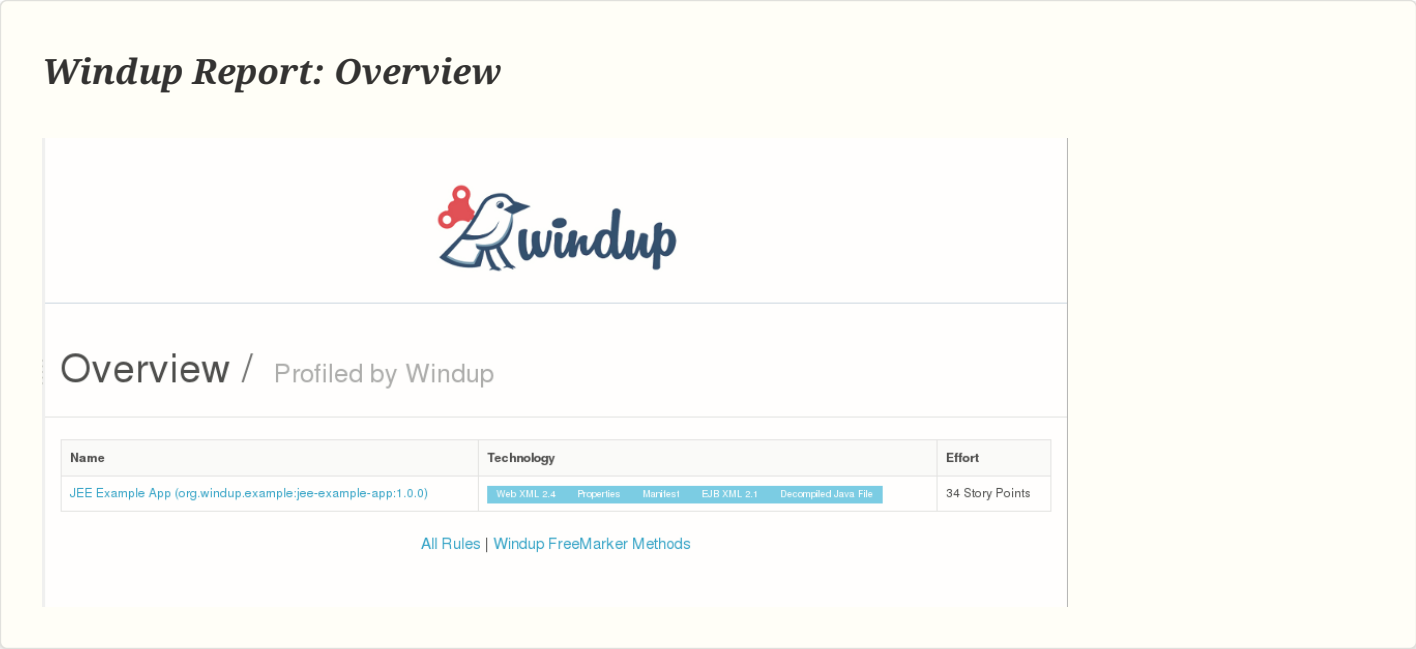
- `index.html`: This is the landing page for the report.
- `archives/`: Contains the archives extracted from the application
- `graph/`: Contains binary graph database files
- `reports/`: This directory contains the generated HTML report files
- `stats/`: Contains Windup performance statistics

The examples below use the test-files/jee-example-app-1.0.0.ear located in the Windup GitHub source repository for input and specify the `com.acme` and `org.apache` package name prefixes to scan. For example:

```
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 --packages com.acme org.apache
```

Open the Report

Use your favorite browser to open the `index.html` file located in the output report directory. You should see something like the following:



This page lists the applications that were processed along with the technologies that were encountered.

Click on the link under the **Name** column to view the Windup application report page.

Report Sections

Application Report Page

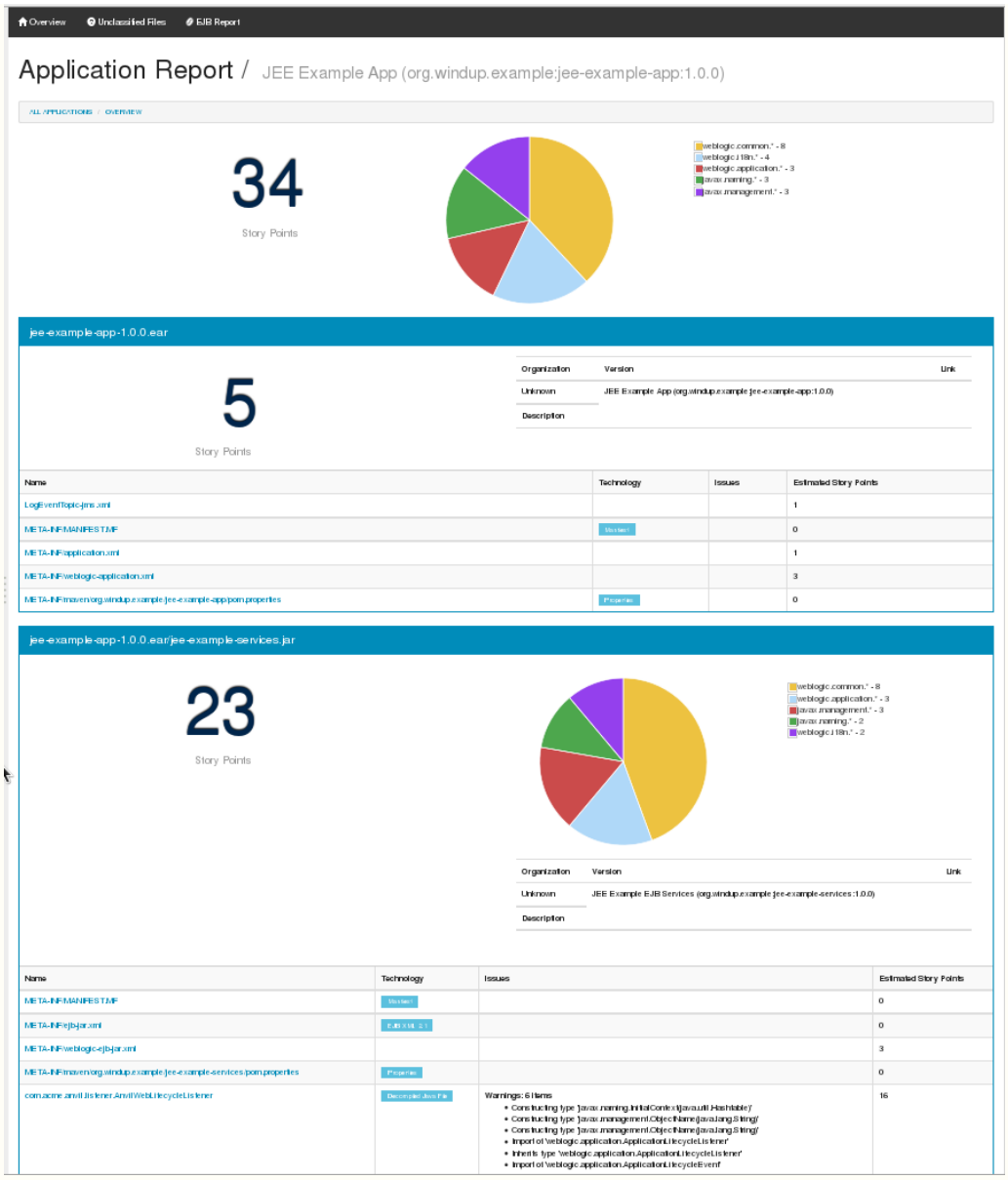
The first section of the application report page summarizes the migration effort. It displays the following information both graphically and in list form by application artifact for each file that is analyzed.

- The file name
- The file type
- A list of issues, if any, that were found in the file
- The estimated total *Story Points* to migrate the file. *Story Points* are covered in more detail in the Windup Rules Development Guide.

> **NOTE** The 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.

In the following Windup Application Report example, the migration of the **JEE Example App** EAR is assigned a total of 34 story points. A pie chart shows the breakdown of story points by package. This is followed by a section for each of the archives contained in the EAR. It provides the total of the story points assigned to the archive and lists the files contained in archive along with the warnings and story point assigned to each file.

Windup Report: Application Report



Archive Analysis Sections

Each archive 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.

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

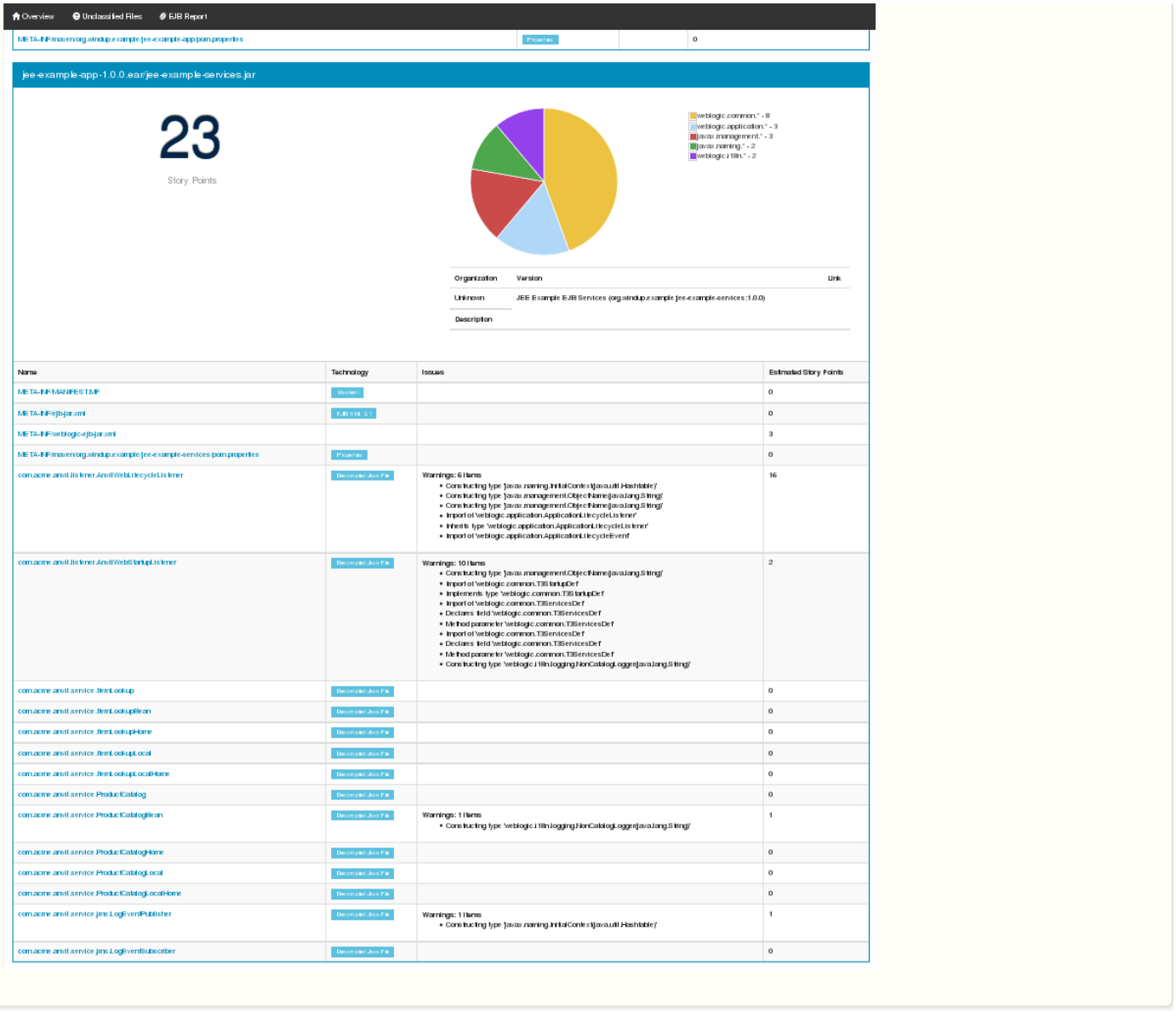
Warnings about areas of code that need review or changes.

Estimated Story Points

Level of effort required for migrating the file.

The following is an example of the archive analysis summary section of a Windup Report. The following is an the analysis of the `WINDUP_SOURCE/test-files/jee-example-app-1.0.0.ear/jee-example-services.jar`.

> ***Windup Report: Application Report (jee-example-app-1.0.0.ear/jee-example-services.jar)***



File Analysis Pages

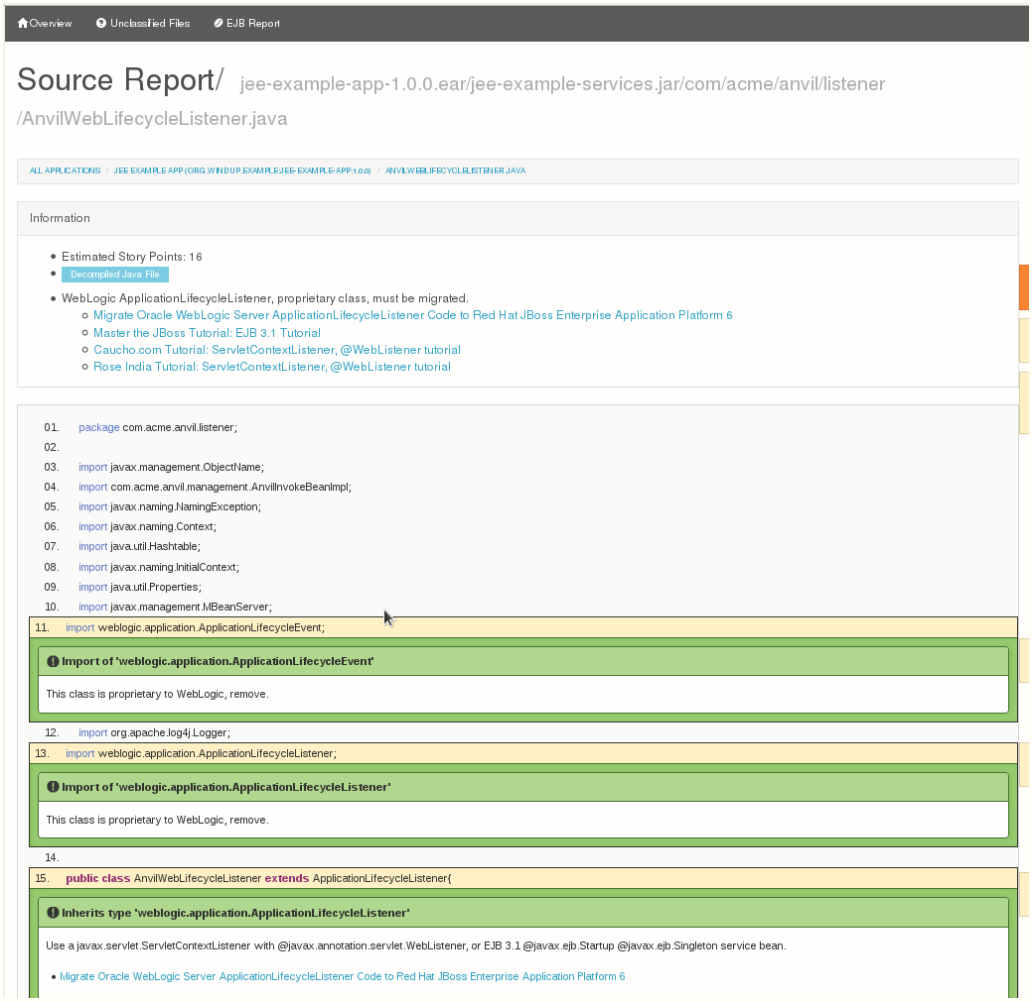
The analysis of the `jee-example-services.jar` lists the files in the JAR and the warnings and story points assigned to each one. Notice the `com.acme.anvil.listener.AnvilWebLifecycleListener` file, at the time of this test, has 6 warnings and is assigned 16 story points. Click on the file to see the 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 the import of `weblogic.application.ApplicationLifecycleEvent` and report that the class is

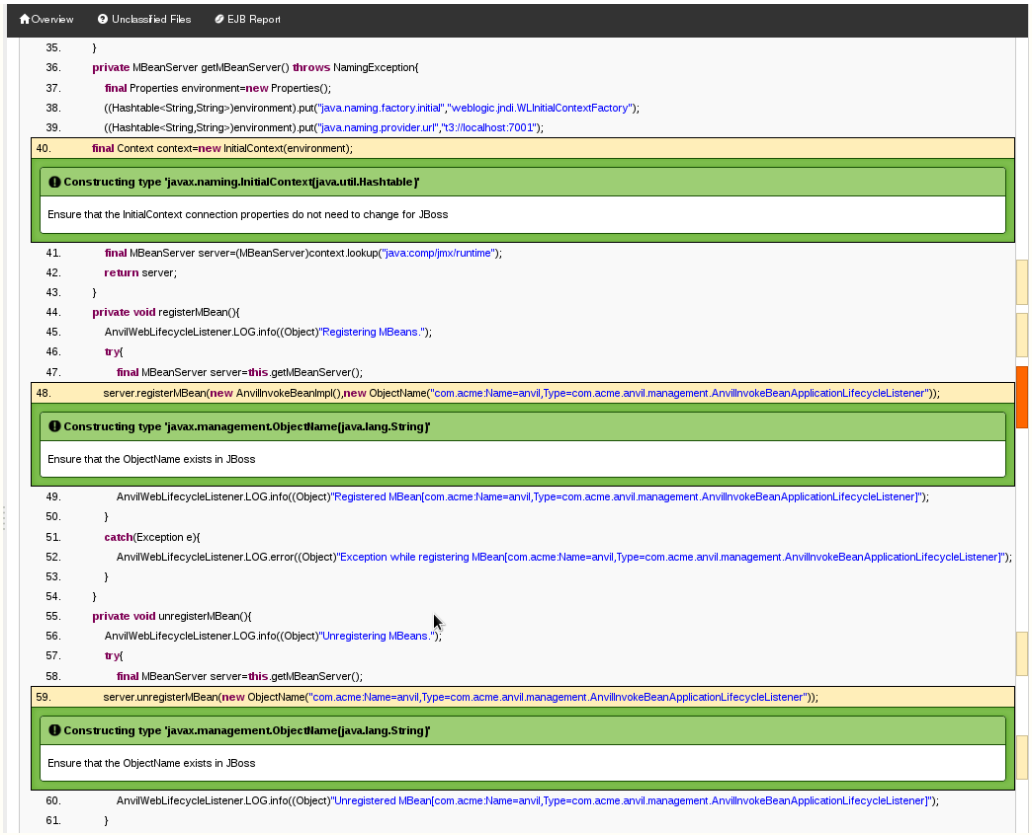
proprietary to WebLogic and must be removed.

Windup Report: Source Report - Part 1



Later in the code, warnings appear for the creation of the InitialContext and for the object name when registering and unregistering an MBeans

Windup Report: Source Report - Part 2



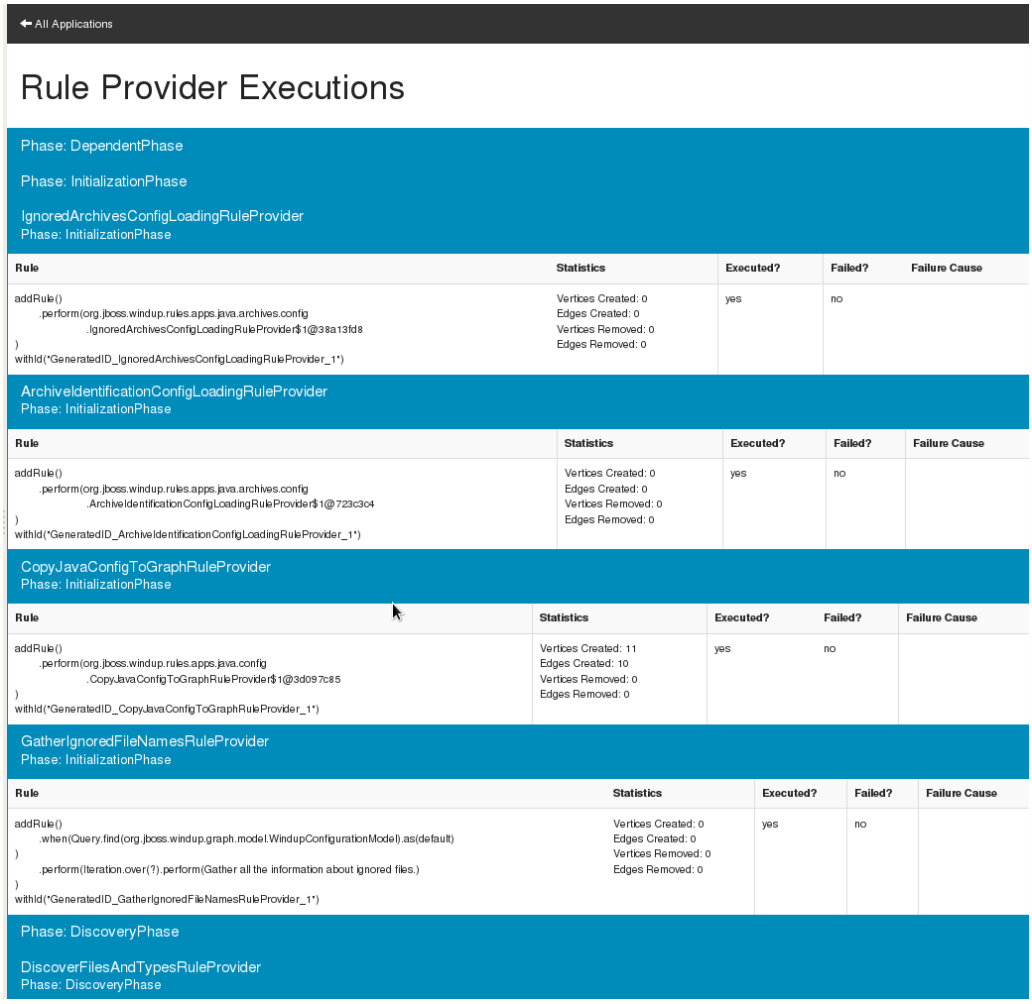
Additional Reports

Explore the Windup `OUTPUT_REPORT_DIRECTORY/reports` folder to find additional reporting information.

Rule Provider Execution Report

The `OUTPUT_REPORT_DIRECTORY/reports/windup_ruleproviders.html` page provides the list of rule providers that executed when running the Windup migration command against the application.

Windup Report: Rule Provider Report



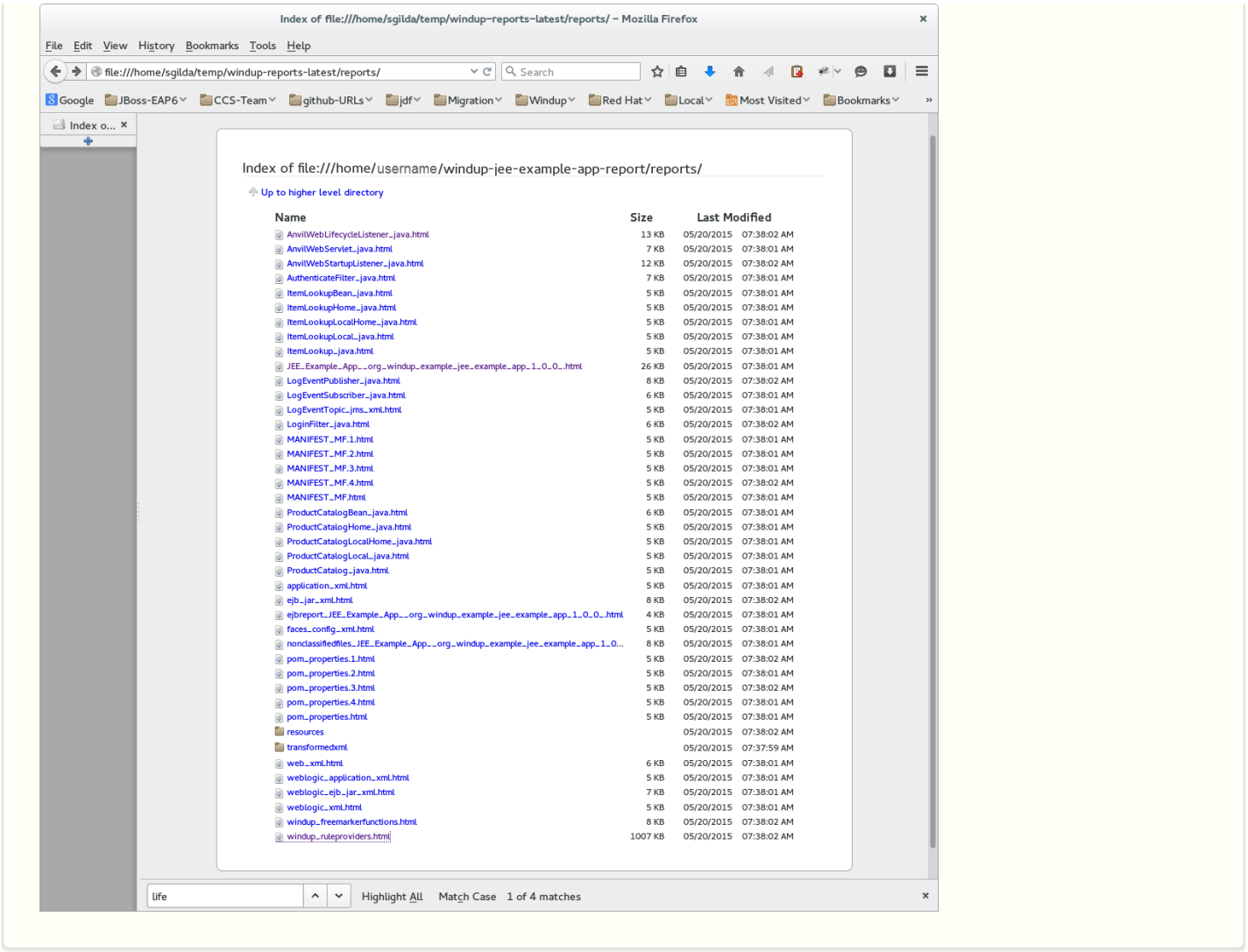
Rule Provider Execution Report

The `OUTPUT_REPORT_DIRECTORY/reports/windup_ruleproviders.html` page provides the list of rule providers that executed when running the Windup migration command against the application.

Individual File Analysis Reports

You can directly access the the file analysis report pages described above by browsing for them by name in the `OUTPUT_REPORT_DIRECTORY/reports/` directory. Because the same common file names can exist in multiple archives, for example "manifest.mf" or "web.xml", Windup adds a unique numeric suffix to each report file name.

Windup Report: Report Directory List



#include::Export-the-Report-for-Use-by-Spreadsheet-Programs[tabsize=4]

Additional Resources

Review the Windup Quickstarts

The Windup quickstarts provide working examples of how to create custom Java-based rule add-ons and XML rules. You can use them as a starting point for creating your own custom rules.

You can download a ZIP file of the latest released version of the quickstarts. Or, if you prefer to play around with the source code, you can fork and clone the windup-quickstarts project repository.

Download the Latest Quickstart ZIP

To download the latest quickstart ZIP file, browse to: https://github.com/windup/windup-quickstarts/releases

Click on the most recent release to download the ZIP to your local file system.

Fork and Clone the Quickstart GitHub Project

If you don't have the GitHub client (`git`), download it from: http://git-scm.com/

1. Click the `Fork` link on the Windup quickstart GitHub page to create the project in your own Git. The forked GitHub repository URL created by the fork should look like this: https://github.com/YOUR_USER_NAME/windup-quickstarts.git
2. Clone your Windup quickstart repository to your local file system:

```
git clone https://github.com/YOUR_USER_NAME/windup-quickstarts.git
```

3. This creates and populates a `windup-quickstarts` directory on your local file system. Navigate to the newly created directory, for example

```
cd windup-quickstarts/
```

4. If you want to be able to retrieve the lates code updates, add the remote `upstream` repository so you can fetch any changes to the original forked repository.

```
git remote add upstream https://github.com/windup/windup-quickstarts.git
```

5. To get the latest files from the `upstream` repository.

```
git reset --hard upstream/master
```

Get Involved

How can you help?

To help us make Windup cover most application constructs and server configurations, including yours, you can help with any of the following items. Many require only a few minutes of your time!

- Send an email to windup-users@lists.jboss.org and let us know what should Windup migration rules 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.
- You can 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.
- You can also 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!

Important Links

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

Known Windup Issues

Windup known issues are tracked here: Open Windup issues

Report Issues with Windup

Windup uses JIRA as its issue tracking system. If you encounter an issue executing Windup, please file a JIRA Issue.

Create a JIRA Account

If you do not yet have a JIRA account, create one using the following procedure.

1. Open a browser to the following URL: https://issues.jboss.org/secure/Dashboard.jspa
2. Click the *Sign Up* link in the top right side of the page.
3. Enter your email address and click the `Confirm address` button.
4. Follow the instructions sent to your email address.

Create a JIRA Issue

1. Open a browser to the following URL: https://issues.jboss.org/secure/CreateIssue!default.jspa.
   - If you have not yet logged in, click the *Log In* link at the top right side of the page.
   - Enter your credentials and click the `LOGIN` button.
   - You are then redirected back to the **Create Issue** page.
2. Choose the following options and click the `Next` button.
   - **Project**: *Windup*
   - **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.

4. Click the `Create` button to create the JIRA issue.
5. 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 by choosing `More → Attach Files`. You are provided with an option to restrict visibility to JBoss employees.

Appendix

Glossary of Terms Used in Windup

Rules Terms

Rule

A piece of code that performs a single unit of work during the migration process. Depending on the complexity of the rule, it may or may not include configuration data. Extensive configuration information may be externalized into external configuration, for example, a custom XML file. The following is an example of a Java-based rule added to the `JDKConfig` RuleProvider class.

```
.addRule()

.when(JavaClass.references("java.lang.ClassLoader$").at(TypeReferenceLocation.TYPE))
    .perform(Classification.as("Java Classloader, must be migrated.")
    .with(Link.to("Red Hat Customer Portal: How to get resources via the ClassLoader
in a JavaEE application in JBoss EAP",
"https://access.redhat.com/knowledge/solutions/239033"))
     .withEffort(1))
```

RuleProvider

An implementation of OCPSoft ConfigurationProvider class specifically for Windup. It provides Rule instances and the relevant RuleProviderMetadata for those Java-based and XML-based Rule instances.

Ruleset

A ruleset is a group of one or more RuleProviders that targets a specific area of migration, for example, `Spring → Java EE 6` or `WebLogic → JBoss EAP`. A ruleset is packaged as a JAR and contains additional information needed for the

migration, such as operations, conditions, report templates, static files, metadata, and relationships to other rulesets. The following Windup projects are rulesets.

- rules-java-ee
- rules-xml

Rules Metadata

Information about whether a particular ruleset applies to a given situation. The metadata can include the source and target platform and frameworks.

Rules Pipeline

A collection of rules that feed information into the knowledge graph.

Reporting Terms

Level of effort

The effort required to complete the migration task.*Level of effort* is represented as *story points* in the Windup reports.

Story Point

A term commonly used in Scrum Agile software development methodology to estimate the *level of effort* needed to implement a feature or change. It does not necessarily translate to man-hours, but the value should be consistent across tasks. Story points are covered in more detail in the Windup Rules Development Guide.

Optimize Windup Performance

Overview

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.

Tips to Optimize Performance

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 the `--target` platform on the on the `WINDUP_HOME/bin/windup` command line to limit the execution of rules to only those that apply to this target platform.
- Be sure to specify a comma-delimited 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` 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.

Hardware Upgrade Suggestions

If the steps above do not improve performance, you may need to upgrade your hardware.

- 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 4-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.

Last updated 2015-08-06 10:50:16 EDT