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.

Windup can be run as a standalone application or as a plug-in to Red Hat JBoss Developer Studio. Running from a [Forge](#) environment, it 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 plugins, rules and reports.

- The data graph organizes the findings to be searchable and queryable.

Extensibility   Windup can be extended by developers, users, and 3rd-parties.

- 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 is based on the skills required and the classification of migration work needed.

- Lift and Shift - The code or file is standards-based and can be ported to the new environment with no changes.

- Mapped - There is a standard mapping algorithm to port the code or file to the new environment.

- Custom – The code or file must be rewritten or modified to work in the new environment.

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.

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

- 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-2.2.0-Final folder extracted from the downloaded ZIP file.

- If you build Windup from GitHub source, WINDUP_HOME refers to the windup-distribution-2.2.0-Final folder extracted from the Windup source dist/target/windup-distribution-2.2.0-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 like the following when you execute Windup.

   ```
   Command: windup-migrate-app was not found
   ```

2. Download the latest Windup ZIP distribution.

3. Extract the ZIP file in to a directory of your choice.

Execute Windup

Prerequisites

Before you begin, you must gather the following information.

1. The fully qualified path of the application archive or folder you plan to migrate.

2. The fully qualified path to a folder that will contain the resulting report information.

   - If the folder does not exist, it is created by Windup.

   - If the folder exists, you are prompted with the message:

     ```
     Overwrite all contents of <OUTPUT_DIRECTORY> (anything already in the directory
     will be deleted)? [y/N]
     ```

     Choose "y" if you want Windup to delete and recreate the directory.

   - If you are confident you want to overwrite the output directory, you can specify `--overwrite` on the command line to automatically delete and recreate the directory.

     > **NOTE** Be careful not to specify a directory that contains important information!

3. You must also provide a list of the application packages to be evaluated.

   - 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.

   - 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.

Start Windup

For information about the use of WINDUP_HOME in the instructions below, see
[About the WINDUP_HOME Variable](#).

1. Open a terminal and navigate to the WINDUP_HOME/bin directory

2. Type the following command to start Windup:

```
For Linux:    windup/bin $ ./windup
For Windows:  C:\WINDUP_HOME\bin> windup
```

3. You are presented with the following prompt.

```
Using Windup at WINDUP_HOME


 _          ___             __
| |      / (_)___   ___/ /_  _____
| | /| / / / __ \/ __  / / / / __ \
| |/ |/ / / / / / /_/ / /_/ / /_/ /
|__/|__/_/_/ /_/\__,_/\__,_/ .___/
                          /_/

JBoss Windup, version [ 2.2.0.Final ] - JBoss, by Red Hat, Inc. [
http://windup.jboss.org ]

[windup-distribution-2.2.0.Final]$
```

Run Windup

1. The command to run Windup is `windup-migrate-app`.

2. This command can take arbitrary options processed by different addons. The list
   of options in the core Windup distribution can be found in [Javadoc](#). Most
   commonly used command line arguments are:

   --input INPUT_ARCHIVE_OR_FOLDER

   This is the fully qualified application archive or source path.

   --output OUTPUT_REPORT_DIRECTORY

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

NOTE	If the **OUTPUT_REPORT_DIRECTORY** directory exists and you do not specify the `--overwrite` argument, you are prompted to overwrite the contents. If you respond "y", it is deleted and recreated by Windup, so be careful not to specify an output directory that contains important information!

--overwrite (optional)

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

--userRulesDirectory

Points to a directory to load XML rules from. (Search pattern: *.windup.groovy and *.windup.xml)

--packages PACKAGE_1, PACKAGE_2, PACKAGE_N (optional)

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

--excludePackages PACKAGE_1, PACKAGE_2, PACKAGE_N (optional)

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

--source-mode true (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`.

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

```
windup-migrate-app --input INPUT_ARCHIVE_OR_FOLDER --output
OUTPUT_REPORT_DIRECTORY --packages PACKAGE_1 PACKAGE_2 PACKAGE_N
```

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

```
windup-migrate-app --sourceMode true --input INPUT_ARCHIVE_OR_FOLDER --output
OUTPUT_REPORT_DIRECTORY --packages PACKAGE_1 PACKAGE_2 PACKAGE_N
```

See [Windup Command Examples](#) below for concrete examples of commands that

use source code directories and archives located in the Windup GitHub repository.

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

```
***SUCCESS*** Windup execution successful!
```

5. To exit Windup, type:

```
exit
```

6. 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
```

7. For details on how to evaluate the report data, see[Review the Report](#).

Run Windup in Batch Mode

Windup can be also executed in batch mode within a shell or batch script using the `--evaluate` argument as follows.

1. Open a terminal and navigate to the WINDUP_HOME directory.

2. Type the following command to run Windup in batch mode:

```
For Linux:     $ bin/windup --evaluate "windup-migrate-app --input INPUT_ARCHIVE
--output OUTPUT_REPORT --packages PACKAGE_1 PACKAGE_2 PACKAGE_N"
For Windows:   > bin\windup.bat --evaluate "windup-migrate-app --input
INPUT_ARCHIVE --output OUTPUT_REPORT --packages PACKAGE_1 PACKAGE_2 PACKAGE_N"
```

Windup Command Line Help

To see the complete list of available arguments for the `windup-migrate-app` command, execute the following command in the Windup prompt:

```
man windup-migrate-app
```

Command Examples

The following command examples report against applications located in the Windup source [test-files](#) directory.

Source Code Example

The following command runs against the [seam-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.

```
windup-migrate-app --sourceMode true --input /home/username/windup-source/test-
files/seam-booking-5.2/ --output /home/username/windup-reports/seam-booking-report --
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.

```
windup-migrate-app --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
```

Windup Batch Example

The following Windup batch 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.

```
For Linux: $ bin/windup --evaluate "windup-migrate-app --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"
For Windows: > bin\windup.bat --evaluate "windup-migrate-app --input \windup-
source\test-files\jee-example-app-1.0.0.ear --output \windup-reports\jee-example-app-
```

```
1.0.0.ear-report --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-migrate-app --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:

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. A *Story Point* is 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.

In the following Windup Application Report example, the migration of the **JEE Example App** EAR is assigned a total of 42 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.

Example of a Windup 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. In this example, it's the analysis of the `WINDUP_SOURCE/test-files/jee-example-app-1.0.0.ear/jee-example-services.jar`.

jee-example-app-1.0.0.ear/jee-example-services.jar

14
Story Points

Legend:
- weblogic.common.* - 8
- javax.management.* - 3
- weblogic.application.* - 2
- javax.naming.* - 2
- weblogic.i18n.* - 2

Name	Technology	Issues	Estimated Story Points
META-INF/MANIFEST.MF	Manifest		0
META-INF/ejb-jar.xml	EJB XML		0
META-INF/weblogic-ejb-jar.xml			3
META-INF/maven/org.windup.example/jee-example-services/pom.properties	Properties		0
com.acme.anvil.listener.AnvilWebLifecycleListener	Decompiled Java File	Warnings: 5 items • Ensure that the InitialContext connection properties do not need to change for JBoss • Ensure that the ObjectName exists in JBoss • Ensure that the ObjectName exists in JBoss • This class is proprietary to Weblogic, remove. • This class is proprietary to Weblogic, remove.	7
com.acme.anvil.listener.AnvilWebStartupListener	Decompiled Java File	Warnings: 10 items • Ensure that the ObjectName exists in JBoss • Replace with EJB 3.1 @Singleton / @Startup annotations. • Replace with EJB 3.1 @Singleton / @Startup annotations. • In JBoss 5, replace with an MBean. • In JBoss 5, replace with an MBean. • In JBoss 5, replace with an MBean. • In JBoss 6, replace with an @Singleton EJB. • In JBoss 6, replace with an @Singleton EJB. • In JBoss 6, replace with an @Singleton EJB. • Requires migration effort	2
com.acme.anvil.service.ProductCatalogBean	Decompiled Java File	Warnings: 1 items • Requires migration effort	1
com.acme.anvil.service.jms.LogEventPublisher	Decompiled Java File	Warnings: 1 items • Ensure that the InitialContext connection properties do not need to change for JBoss	1

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 has 5 warnings and is assigned 7 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.

Source Report / jee-example-app-1.0.0.ear/jee-example-services.jar/com/acme

/anvil/listener/AnvilWebLifecycleListener.java

Information

- Estimated Story Points: 7
- Decompiled Java File

```
01.    package com.acme.anvil.listener;
02.
03.    import java.lang.Class;
04.    import java.lang.Exception;
05.    import javax.management.ObjectName;
06.    import com.acme.anvil.management.AnvilInvokeBeanImpl;
07.    import javax.naming.NamingException;
08.    import javax.naming.Context;
09.    import java.util.Hashtable;
10.    import javax.naming.InitialContext;
11.    import java.util.Properties;
12.    import javax.management.MBeanServer;
13.    import java.lang.Object;
14.    import java.lang.StringBuilder;
15.    import weblogic.application.ApplicationLifecycleEvent;
```
> This class is proprietary to Weblogic, remove.
```
16.    import java.lang.String;
17.    import org.apache.log4j.Logger;
18.    import weblogic.application.ApplicationLifecycleListener;
```
> This class is proprietary to Weblogic, remove.
```
19.
20.    public class AnvilWebLifecycleListener extends ApplicationLifecycleListener{
21.        private static Logger LOG;
22.        private static final String MBEAN_NAME="com.acme:Name=anvil,Type=com.acme.anvil.management.AnvilInvokeBeanApplicationLifecycleListener";
23.        public void preStart(final ApplicationLifecycleEvent evt){
24.            final String appName=evt.getApplicationContext().getApplicationName();
25.            AnvilWebLifecycleListener.LOG.info((Object)("Before Start Application["+appName+"]"));
26.        }
27.        public void postStart(final ApplicationLifecycleEvent evt){
```

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

```
41.    private MBeanServer getMBeanServer() throws NamingException{
42.        final Properties environment=new Properties();
43.        ((Hashtable<String,String>)environment).put("java.naming.factory.initial","weblogic.jndi.WLInitialContextFactory");
44.        ((Hashtable<String,String>)environment).put("java.naming.provider.url","t3://localhost:7001");
45.        final Context context=new InitialContext(environment);
```

> Ensure that the InitialContext connection properties do not need to change for JBoss

```
46.        final MBeanServer server=(MBeanServer)context.lookup("java:comp/jmx/runtime");
47.        return server;
48.    }
49.    private void registerMBean(){
50.        AnvilWebLifecycleListener.LOG.info((Object)"Registering MBeans.");
51.        try{
52.            final MBeanServer server=this.getMBeanServer();
53.            server.registerMBean(new AnvilInvokeBeanImpl(),new ObjectName("com.acme:Name=anvil,Type=com.acme.anvil.management.AnvilInvokeBeanApp
```

> Ensure that the ObjectName exists in JBoss

```
54.            AnvilWebLifecycleListener.LOG.info((Object)"Registered MBean[com.acme:Name=anvil,Type=com.acme.anvil.management.AnvilInvokeBeanApplic
55.        }
56.        catch(Exception e){
57.            AnvilWebLifecycleListener.LOG.error((Object)"Exception while registering MBean[com.acme:Name=anvil,Type=com.acme.anvil.management.Anvil
58.        }
59.    }
60.    private void unregisterMBean(){
61.        AnvilWebLifecycleListener.LOG.info((Object)"Unregistering MBeans.");
62.        try{
63.            final MBeanServer server=this.getMBeanServer();
64.            server.unregisterMBean(new ObjectName("com.acme:Name=anvil,Type=com.acme.anvil.management.AnvilInvokeBeanApplicationLifecycleListener"
```

> Ensure that the ObjectName exists in JBoss

```
65.            AnvilWebLifecycleListener.LOG.info((Object)"Unregistered MBean[com.acme:Name=anvil,Type=com.acme.anvil.management.AnvilInvokeBeanApp
66.        }
67.        catch(Exception e){
68.            AnvilWebLifecycleListener.LOG.error((Object)"Exception while unregistering MBean[com.acme:Name=anvil,Type=com.acme.anvil.management.An
69.    }
```

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.

Rule Provider Executions

<- All Applications

org.jboss.windup.rules.apps:rules-java.CopyJavaConfigToGraphRuleProvider
Phase: Pre Discovery

Rule	Statistics	Executed?	Failed?	Failure Cause
addRule() .perform(org.jboss.windup.rules.apps.java.config .CopyJavaConfigToGraphRuleProvider$1@71d182d7) .withId("GeneratedID_org.jboss.windup.rules.apps.rules-java.CopyJavaConfigToGraphRuleProvider_1")	Vertices Created: 3 Edges Created: 2 Vertices Removed: 0 Edges Removed: 0	yes	no	

org.jboss.windup.rules.apps:rules-java.DiscoverFileTypesRuleProvider
Phase: Discovery

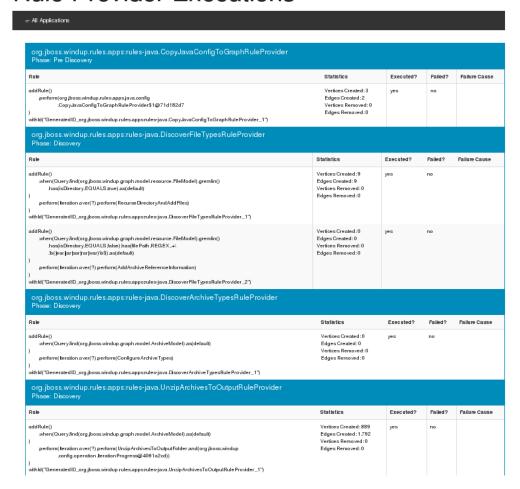| Rule	Statistics	Executed?	Failed?	Failure Cause
addRule() .when(Query.find(org.jboss.windup.graph.model.resource.FileModel).gremlin() .has(isDirectory,EQUALS,true).as(default)) .perform(Iteration.over(?).perform(RecurseDirectoryAndAddFiles)) .withId("GeneratedID_org.jboss.windup.rules.apps.rules-java.DiscoverFileTypesRuleProvider_1")	Vertices Created: 9 Edges Created: 9 Vertices Removed: 0 Edges Removed: 0	yes	no	
addRule() .when(Query.find(org.jboss.windup.graph.model.resource.FileModel).gremlin() .has(isDirectory,EQUALS,false).has(filePath,REGEX,.+\ .\b(|ear|jar|sar|rar|war)'\b$).as(default)) .perform(Iteration.over(?).perform(AddArchiveReferenceInformation)) .withId("GeneratedID_org.jboss.windup.rules.apps.rules-java.DiscoverFileTypesRuleProvider_2")	Vertices Created: 0 Edges Created: 0 Vertices Removed: 0 Edges Removed: 0	yes	no	

org.jboss.windup.rules.apps:rules-java.DiscoverArchiveTypesRuleProvider
Phase: Discovery

Rule	Statistics	Executed?	Failed?	Failure Cause
addRule() .when(Query.find(org.jboss.windup.graph.model.ArchiveModel).as(default)) .perform(Iteration.over(?).perform(ConfigureArchiveTypes)) .withId("GeneratedID_org.jboss.windup.rules.apps.rules-java.DiscoverArchiveTypesRuleProvider_1")	Vertices Created: 0 Edges Created: 0 Vertices Removed: 0 Edges Removed: 0	yes	no	

org.jboss.windup.rules.apps:rules-java.UnzipArchivesToOutputRuleProvider
Phase: Discovery

Rule	Statistics	Executed?	Failed?	Failure Cause
addRule() .when(Query.find(org.jboss.windup.graph.model.ArchiveModel).as(default)) .perform(Iteration.over(?).perform(UnzipArchivesToOutputFolder.and(org.jboss.windup .config.operation.IterationProgress@4061a2cd))) .withId("GeneratedID_org.jboss.windup.rules.apps.rules-java.UnzipArchivesToOutputRuleProvider_1")	Vertices Created: 889 Edges Created: 1,792 Vertices Removed: 0 Edges Removed: 0	yes	no	

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.

Index of file:///home/username/windup-jee-example-app-report/reports/

⬆ Up to higher level directory

Name	Size	Last Modified
Agent_java.1.html	7 KB	10/31/2014 08:41:14 AM
Agent_java.2.html	7 KB	10/31/2014 08:41:15 AM
AnvilWebLifecycleListener_java.1.html	11 KB	10/31/2014 08:41:14 AM
AnvilWebStartupListener_java.1.html	11 KB	10/31/2014 08:41:14 AM
AppenderDynamicMBean_java.1.html	16 KB	10/31/2014 08:41:14 AM
AppenderDynamicMBean_java.2.html	16 KB	10/31/2014 08:41:14 AM
AuthenticateFilter_java.1.html	7 KB	10/31/2014 08:41:14 AM
HierarchyDynamicMBean_java.1.html	15 KB	10/31/2014 08:41:14 AM
HierarchyDynamicMBean_java.2.html	15 KB	10/31/2014 08:41:14 AM
JDBCAppender_java.1.html	10 KB	10/31/2014 08:41:14 AM
JDBCAppender_java.2.html	10 KB	10/31/2014 08:41:14 AM
JEE_Example_App__org_windup_example_jee_example_app_1_0_0_.1.html	26 KB	10/31/2014 08:41:16 AM
JMSAppender_java.1.html	14 KB	10/31/2014 08:41:15 AM
JMSAppender_java.2.html	14 KB	10/31/2014 08:41:14 AM
LogEventPublisher_java.1.html	8 KB	10/31/2014 08:41:15 AM
LogEventTopic_jms_xml.1.html	5 KB	10/31/2014 08:41:15 AM
LoggerDynamicMBean_java.1.html	14 KB	10/31/2014 08:41:15 AM
LoggerDynamicMBean_java.2.html	14 KB	10/31/2014 08:41:14 AM
LoginFilter_java.1.html	6 KB	10/31/2014 08:41:15 AM
MANIFEST_MF.1.html	5 KB	10/31/2014 08:41:16 AM
MANIFEST_MF.2.html	5 KB	10/31/2014 08:41:15 AM
MANIFEST_MF.3.html	5 KB	10/31/2014 08:41:15 AM

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. The quickstarts are available on GitHub here: https://github.com/windup/windup-quickstarts

You can fork and clone the project to have access to regular updates or you can download a ZIP file of the latest version.

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 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.

- 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

See the [list of links to Windup resources](#).

Important Links

- Windup wiki: https://github.com/windup/windup/wiki

- Windup documentation (generated from the Wiki documentation at the link above):

  - [Windup User Guide](#)

  - [Windup Rules Development Guide](#)

  - [Windup Core Development Guide](#)

  - [Windup Javadoc](#)

- Windup forums: https://community.jboss.org/en/windup

  - Windup 0.x legacy forums: https://developer.jboss.org/en/windup

- Windup issue tracker: https://issues.jboss.org/browse/WINDUP

- Windup users mailing List: windup-users@lists.jboss.org

- Windup developers mailing list: windup-dev@lists.jboss.org

- Windup commits mailing list: windup-commits@lists.jboss.org

- Windup on Twitter: @JBossWindup

- Windup IRC channel: Server FreeNode (`irc.freenode.net`), channel `#windup`.

  - Windup IRC Chat transcripts: http://bit.ly/windup-transcripts

  - Windup meeting IRC Chat transcripts: transcripts: http://bit.ly/windup-meetings

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 level of effort required for the migration task. The following values are used in the reports.

- *Lift and Shift*: The code or file is standards-based and can be ported to the new environment with no changes.

- *Known Solution*: There is a standard mapping algorithm to port the code or file to the new environment.

- *Custom*: The code or file must be rewritten or modified to work in the new environment.

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.

Rule Story Points

What are Story Points?

Story points are an abstract metric commonly used in Scrum Agile software development methodology to estimate the level of effort needed to implement a feature or change. They are based on a [modified Fibonacci sequence](#).

In a similar manner, Windup uses story points to express the level of effort needed to migrate particular application constructs, and in a sum, the application as a whole. It does not necessarily translate to man-hours, but the value should be consistent across tasks.

How Story Points are Estimated in Rules

Estimating 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
Lift and Shift	0	The code or file is standards-based and requires no effort.
Mapped	1- 2 per file	There is a standard mapping algorithm to port the code or file. The number of story points required is small, but is dependent on the number of files to port.

Level of Effort	Story Points	Description
Custom	5 - 20 per change or component	The number of story points required to modify and rewrite code depends on the complexity of the change, the number of unknown imports, the size of the files, and the number of components. The following are examples of how to estimate story points. • Port MyBatis to JPA: '20' story points per query. • Port a web page from one web framework to another depends on the complexity and the number of components involved in the migration. You could estimate '20' story points per component.