Oracle® Fusion Middleware

SmartUpgrade User's Guide

11*g* Release 2 (11.1.2.1.0)

E15878-03

September 2011

ORACLE®

Oracle Fusion Middleware SmartUpgrade User's Guide, 11*g* Release 2 (11.1.2.1.0)

E15878-03

Primary Author: Shynitha K S

Contributors: Brian Lininger, Cecilia Broadaway, Clyde Iwamoto, Frances Zhao, Gavin Steyn, Jason Minton, John Bracken, Lixin Zheng, Michael Rubino, Nilesh Junnarkar, Olga Peschansky, Pascal Filion, Peter LaQuerre, Reza Shafii, Robert St. Jean, Roger Striffler, Sitaraman Swaminathan, Showvik Roy Chowdhuri, Velmurugan Subramanian, Wendy Puckett

Contents

3 Using the SmartUpgrade Command Line

4 Using SmartUpgrade Generated Artifacts

Preface

This preface contains the following sections:

Audience

This manual is intended for Java EE application developers who are upgrading applications previously deployed on Oracle Containers for Java EE applications and who want to redeploy the applications on Oracle WebLogic Server and Oracle Fusion Middleware 11*g*.

Documentation Accessibility

For information about Oracles commitment to accessibility, visit the Oracle Accessibility Program website at
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

For more information, see the following related documentation available in the Oracle Fusion Middleware 11*g* documentation library:

- Related Upgrade Documentation
  - *Oracle Fusion Middleware Upgrade Planning Guide*
  - *Oracle Fusion Middleware Upgrade Guide for Java EE*
- *Oracle Fusion Middleware Installation Planning Guide*
- *Oracle Fusion Middleware Administrator's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to Oracle WebLogic Server SmartUpgrade

This chapter introduces the Oracle WebLogic Server SmartUpgrade software. It includes the following sections:

1.1 What Is Oracle WebLogic Server SmartUpgrade?

Oracle WebLogic Server SmartUpgrade is an Oracle JDeveloper extension and command-line utility that analyzes the applications you deployed previously on OC4J. It then offers advice and performs actions that can help you successfully redeploy the applications on Oracle WebLogic Server.

You can analyze an application archive, or you can analyze an application or project you have opened in Oracle JDeveloper. In addition, SmartUpgrade can analyze the OC4J server where you deployed your applications and provide advice on how to set up a similar configuration in Oracle WebLogic Server.

In addition to providing a comprehensive report with specific findings about each application, SmartUpgrade automates some of the upgrade tasks. For example, SmartUpgrade can generate specific types of application artifacts and automatically package the artifacts as an enterprise archive. You can use this archive as a starting point for deploying your application on Oracle WebLogic Server.

1.2 Using SmartUpgrade As Part of Your Oracle Fusion Middleware Upgrade

Oracle WebLogic Server SmartUpgrade is one of several Oracle software tools that can help you upgrade your entire Oracle Application Server or Oracle WebLogic Server environment to Oracle Fusion Middleware 11*g*.

As a result, you can use SmartUpgrade as part of an overall approach to upgrading your environment. Oracle recommends that you use SmartUpgrade in conjunction with the other upgrade-related Oracle Fusion Middleware documentation resources.

Refer to the following for more information:

- For information about upgrading all the components of your application server and middleware environment, refer to the *Oracle Fusion Middleware Upgrade Planning Guide*.
- For specific information about upgrading your Java EE applications and Oracle Application Server 10*g* environment to Oracle WebLogic Server, see the *Oracle Fusion Middleware Upgrade Guide for Java EE*.

1.3 About the SmartUpgrade Report

After SmartUpgrade analyzes a selected enterprise archive or Oracle JDeveloper application or project, SmartUpgrade generates an application upgrade report. You can then browse the report to identify potential issues to address before you can deploy it on Oracle WebLogic Server. Each issue is referred to as a "finding."

Each finding is assigned various attributes, including the level of priority and complexity. The priority and complexity attributes can help you plan the work required to modify and redeploy an application on Oracle WebLogic Server.

For more information, see Section 2.4, "Using a SmartUpgrade Report".

1.4 About SmartUpgrade Artifact Generation

In addition to the upgrade report, SmartUpgrade can optionally generate specific types of application artifacts, such as Oracle WebLogic Server deployment descriptors, data source configurations, Web services, and EJB artifacts.

Instead of reviewing the report findings and creating these artifacts yourself, you can use SmartUpgrade to generate the artifacts for you, which can save you time and effort when upgrading your applications for Oracle WebLogic Server.

For more information, see the following sections:

- Generation of Deployment Descriptors
- Generation of Web Services Artifacts
- Generation of EJB Artifacts

1.4.1 Generation of Deployment Descriptors

If you configure SmartUpgrade to generate artifacts for a typical Java EE application, then SmartUpgrade analyzes the OC4J deployment descriptors within the application and, for specific elements within the OC4J-specific deployment descriptor, generates sample files that contain the equivalent deployment descriptor elements that can be used to deploy the application on Oracle WebLogic Server.

You can then use the generated deployment descriptors as a starting point for creating the require Oracle WebLogic Server deployment descriptors for your upgraded application.

Besides reviewing the upgrade documentation and reviewing the advice available in the upgrade report findings, you can also configure SmartUpgrade to generate a specific set of Oracle WebLogic Server Web application deployment descriptor elements for you.

Specifically, if you configure SmartUpgrade to generate artifacts, and the software identifies an `orion-web.xml` file within the application, SmartUpgrade

automatically generates the equivalent Oracle WebLogic Server deployment descriptor elements in a sample weblogic.xml file:

- virtual-directory
- resource-eve-ref-mapping
- ejb-ref-mapping
- default-char-set
- jsp-print-nulls
- default-mime-type
- directory-browsing
- persistence-path
- session-tracking
- expiration-setting

If your application uses the `session-tracking` and `expiration-setting` elements, then SmartUpgrade generates wrapper Java classes that are automatically included in the archive file that is generated by SmartUpgrade.

For more information, see Chapter 4, "Using SmartUpgrade Generated Artifacts" and Appendix A, "Output Directories Generated by SmartUpgrade".

1.4.2 Generation of Web Services Artifacts

SmartUpgrade makes it possible to quickly and efficiently upgrade and deploy your Web services to Oracle WebLogic Server, while preserving the existing URLs and interfaces with remote applications in your environment.

When you configure SmartUpgrade to generate artifacts for a Web services application, SmartUpgrade analyzes the OC4J Web services and performs three distinct tasks automatically:

1. Uses Oracle Weblogic Server service generation tools to generate the service skeleton artifacts for each WSDL in the application.

   The skeleton includes:

   - A new set of value types or data transfer objects
   - A new service endpoint interface (SEI) and a skeleton Web service
   - Oracle WebLogic Server deployment descriptors

2. Provides the implementation of the skeleton Web service. This implementation is called "glue code" and is generated with required annotations supported by Oracle WebLogic Server.
3. The "glue code" in the generated skeleton Web service dispatches requests to the original Web service implementation class, which is part of the OC4J application being upgraded. It performs this task by converting the new value types into the original value types.

SmartUpgrade ensures that the existing clients of the original OC4J Web services are not affected by the upgrade. The original WSDL contract is preserved and existing clients can continue to interoperate with the new Web service generated for Oracle WebLogic Server.

Note that the generated Web services glue code contains the required annotations.

SmartUpgrade supports the upgrade of the following Web services features:

- Stateless and Stateful Services
- EJB 2.0 and 3.0 Web Services
- RPC-Lit and Doc-LIt Web Services
- Web Service JAX-RPC Client (a stub-based client generated by OC4J WSA utility)
- SOAP Attachments implemented via JAX-RPC handlers
- SOAP Message Transmission Optimization Mechanism (MTOM)
- WS Security

Note: OC4J supports both client java and load services. Note that Oracle WebLogic Server supports only client java. If you try to call load services in Oracle WebLogic Server, then the following error message is displayed:

```
java.lang.Error: JAX-RPC 1.1 method is not supported
in WLS 8.1 clients. If you are attempting to run an
OC4J 10.1.3 JAX-RPC client in WLS, please see the
Web Service Migration Guide for instructions.

at
weblogic.webservice.core.rpc.ServiceFactoryImpl.load
Service(ServiceFactoryImpl.java:65)
```

If you are using JAX-RPC API such as loadService, you should change the code to use the wsa.jar generated Client class, as shown in the following example:

Change

```
DocLitLogger service = (DocLitLogger) m_
factory.loadService ( new URL(m_
serviceURL),DocLitLogger.class, null);

LoggingFacilityLogPortType port =
service.getDocLitLoggerPort();
```

to

```
DocLitLoggerPortClient  client =  new
DocLitLoggerPortClient();

LoggingFacilityLogPortType port = client.getPort();
```

For more information, see Chapter 4, "Using SmartUpgrade Generated Artifacts".

1.4.3 Generation of EJB Artifacts

If your application contains Enterprise Java Beans (EJBs), then SmartUpgrade analyzes your application and attempts to generate the equivalent artifacts that can be deployed on Oracle WebLogic Server.

Specifically, SmartUpgrade generates EJB artifacts for OC4J Native CMP applications, as well as other EJB types, with the following exceptions:

- Auto-generated primary keys are partially supported. SmartUpgrade defaults to Oracle sequence as part of the migration, but Oracle recommends that you manually configure auto-generated keys, as required.
- Generation of EJB artifacts for TopLink CMP applications is not supported. TopLink BMP or Weblogic CMP are recommended; however, the upgrade of these elements must be performed manually by the user.
- There are some EJB features which are not transferable between platforms. Refer to the SmartUpgrade upgrade report for details about the features that are not upgraded for a given application.
- Only the upgrade of OC4J Native message-driven beans (MDBs) is supported.

1.5 SmartUpgrade Oracle JDeveloper Integration and Command-Line Interface

SmartUpgrade is available as an Oracle JDeveloper extension that can be installed using the Oracle JDeveloper **Check for Updates** feature. It is also available as a command-line tool.

Refer to the following resources for details:

- For information on using the Oracle JDeveloper extension, see Chapter 2, "Using SmartUpgrade with Oracle JDeveloper".
- For information about using the command-line interface, see Chapter 3, "Using the SmartUpgrade Command Line"

1.6 Downloading and Installing SmartUpgrade

Refer to the following sections for more information about downloading and installing SmartUpgrade:

1.6.1 Obtaining the SmartUpgrade ZIP Files

You can obtain SmartUpgrade in one of two ways:

- From the Oracle Fusion Middleware Companion CD-ROM, which is part of the Oracle Fusion Middleware CD-ROM pack
- From the Oracle Technology Network (OTN)

For example, to download and unpack the SmartUpgrade ZIP file from OTN:

1. Use your Web browser to navigate to the following URL on OTN:

   ```
   http://www.oracle.com/technology/products/middleware/upgrade/index.html
   ```

   Note that the first time you access OTN, you will have to register. Registration is free and provides you with access to all the resources on OTN.

2. Click the link to download Oracle WebLogic Server SmartUpgrade.

At the time this document was published, the SmartUpgrade download link was located under the Free Downloads image on the right side of the page.

3. Follow the instructions on the screen to download a ZIP file that contains all the required SmartUpgrade files.
4. Unpack the ZIP file in a temporary directory on your local disk.

   Table 1–1 describes the contents of the ZIP file.

Table 1–1 Contents of the SmartUpgrade ZIP File

File	Description
readme.txt	A text file that describes the files in the ZIP file.
smartupgrade.zip	A ZIP file containing the files required to use the SmartUpgrade command-line interface.
releasenotes.txt	A text file that describes late-breaking information about this SmartUpgrade download.
jdeveloper_smartupgrade.zip	A ZIP file containing the files required to install the SmartUpgrade Oracle JDeveloper extension.

1.6.2 Installing the Oracle JDeveloper SmartUpgrade Extension

To install the Oracle JDeveloper SmartUpgrade extension from the contents of the downloadable ZIP file:

1. Verify that you are currently running Oracle JDeveloper 11*g*.

   If necessary, download and install Oracle JDeveloper 11*g* from the following location on OTN:

   ```
   http://www.oracle.com/technology/products/jdev/index.html
   ```

   For Oracle JDeveloper installation instructions, refer to the *Oracle Fusion Middleware Installation Guide for Oracle JDeveloper*.
2. Start Oracle JDeveloper.
3. Select **Check for Updates** from the **Help** menu.
4. Click **Next** on the Welcome page of the Check for Updates wizard.
5. On the Source page, select **Install From Local File**.
6. Click **Browse** to locate and select the `jdeveloper_smartupgrade.zip` file you obtained and unpacked in Section 1.6.1, "Obtaining the SmartUpgrade ZIP Files".
7. Click **Finish** to install the SmartUpgrade extension.
8. Restart Oracle JDeveloper.

   After you restart Oracle JDeveloper, refer to Chapter 2, "Using SmartUpgrade with Oracle JDeveloper".

Note: Future updates to Oracle JDeveloper will be available by selecting one of the **Search Update Centers** options in the Check for Updates wizard as they are available.

1.6.3 Installing the SmartUpgrade Command-Line Interface

When you install the SmartUpgrade Oracle JDeveloper extension, the files required to run the command-line interface are installed automatically.

To verify that the command-line interface has been installed, locate the `smartupgrade.jar` file in the following directory after you install the SmartUpgrade extension:

MW_HOME`\jdeveloper\jdev\extensions\oracle.jdeveloper.smartupgrade.weblogic\`

If you do not install the Oracle JDeveloper extension, and you want to install only the SmartUpgrade command-line interface, you can do so from the contents of the downloadable ZIP file or the SmartUpgrade files available on the Oracle Fusion Middleware 11*g* Release 1 (11.1.1.2.0) Companion CD-ROM.

To install and configure the SmartUpgrade command-line interface without using Oracle JDeveloper:

1. Verify that you have installed and configured one of the following prerequisites:
   - Java 2 Standard Edition or Enterprise Edition Version 1.6 or later

     For more information, refer to the following Web site for information on downloading the Java 1.6 Software Development Kit (SDK) or Java Runtime (JRE):

     `http://www.oracle.com/technetwork/java/index.html`

   - Apache Ant Version 1.7 or later

     Apache Ant 1.7 is available is installed as part of any Oracle WebLogic Server 11*g* installation. You can also download it from the following URL:

     `http://ant.apache.org/`

2. Unpack the `smartupgrade.zip` file into a permanent directory where you will run the program.

   Oracle includes the `smartupgrade.zip` file as part of the SmartUpgrade download from the Oracle Technology Network (OTN), as well as on the Oracle Fusion Middleware 11*g* Release 1 (11.1.1.2.0) Companion CD-ROM.

3. Verify that the following files are unpacked into the directory:
   - `smartupgrade.jar`
   - `Manifest.mf`
   - `readme.txt`
4. For get started using the command-line interface, refer to Chapter 3, "Using the SmartUpgrade Command Line".

1.6.4 Obtaining the Latest Documentation

For the most up-to-date information about SmartUpgrade, including tutorials, data sheet, and the most recent version of this guide, refer to the Oracle Fusion Middleware Upgrade page on the Oracle Technology Network (OTN):

`http://www.oracle.com/technetwork/middleware/upgrade-092995.html`

2

Using SmartUpgrade with Oracle JDeveloper

This chapter describes how to use SmartUpgrade within Oracle JDeveloper. Refer to the following sections for more information:

- Installing and Configuring Oracle JDeveloper
- Verifying the Oracle JDeveloper SmartUpgrade Extension Installation
- Starting and Using the Java EE Upgrade Wizard
- Using a SmartUpgrade Report

2.1 Installing and Configuring Oracle JDeveloper

To use SmartUpgrade with Oracle JDeveloper, you must install Oracle JDeveloper 11*g* Release 2 (11.1.2.1.0) or later.

When you install Oracle JDeveloper 11*g*, you must also install the Oracle WebLogic Server software that is part of the Oracle JDeveloper installation package. SmartUpgrade automatically uses required libraries and other software from the Oracle WebLogic Server software that you install with Oracle JDeveloper.

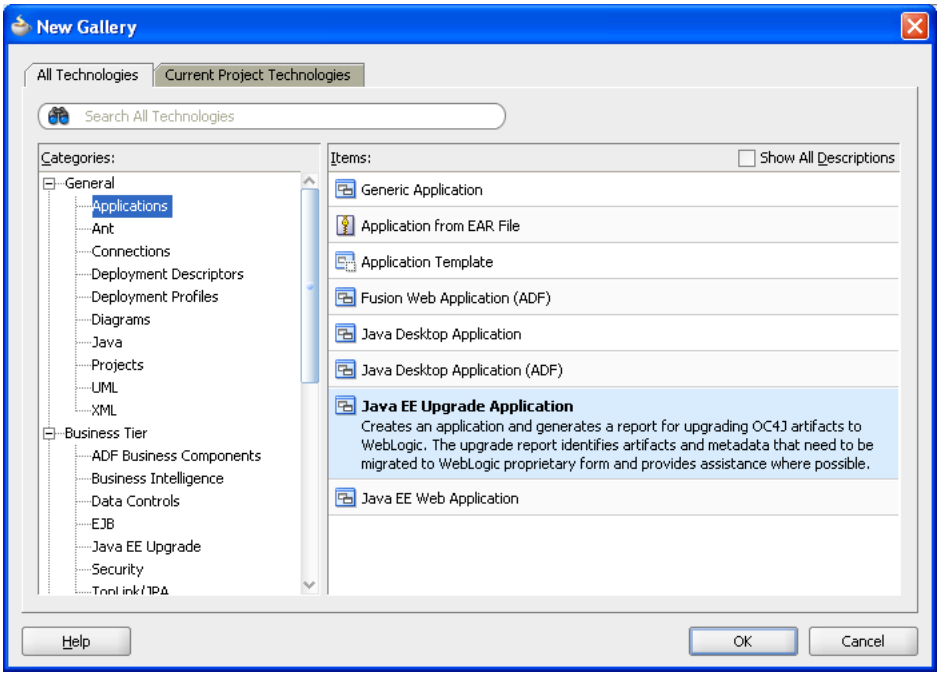
2.2 Verifying the Oracle JDeveloper SmartUpgrade Extension Installation

Before you begin, verify that the Oracle JDeveloper SmartUpgrade extension has been installed and configured:

1. Start Oracle JDeveloper.
2. Select **File** > **New**.
3. In the Categories tree of the New Gallery, select **General** then **Applications**.
4. In the Items list, verify that the **Java EE Upgrade Application** option appears in the list of Items, as shown in Figure 2–1.

   If this option does not appear in the list, then refer to Section 1.6, "Downloading and Installing SmartUpgrade".

Figure 2–1 Java EE Upgrade Application Item in the New Gallery Dialog



2.3 Starting and Using the Java EE Upgrade Wizard

To generate a SmartUpgrade report or to optionally generate artifacts to help you upgrade your applications, you use the Java EE Upgrade Wizard.

Refer to the following sections for more information:

- Starting the Java EE Upgrade Wizard
- Using the Java EE Upgrade Wizard
- Setting Properties on the Required Inputs Page
- Continuing with the Upgrade When Generating Artifacts

2.3.1 Starting the Java EE Upgrade Wizard

There are two ways to start Java EE Upgrade wizard, which guides you through the process of analyzing your application with SmartUpgrade.

The following sections describe these two options:

- Using the Java EE Upgrade Application Wizard
- Using the Java EE Upgrade Project Wizard

2.3.1.1 Using the Java EE Upgrade Application Wizard

To start the Java EE Upgrade wizard and save the results to a new Oracle JDeveloper application:

1. Choose **File** > **New** to display the New Gallery.
2. In the Categories tree, select **General** then **Applications**.
3. In the Items list, double-click **Java EE Upgrade Application** (Figure 2–1).

2.3.1.2 Using the Java EE Upgrade Project Wizard

To start the Java EE Upgrade wizard and save the results to a new project within an existing Oracle JDeveloper application:

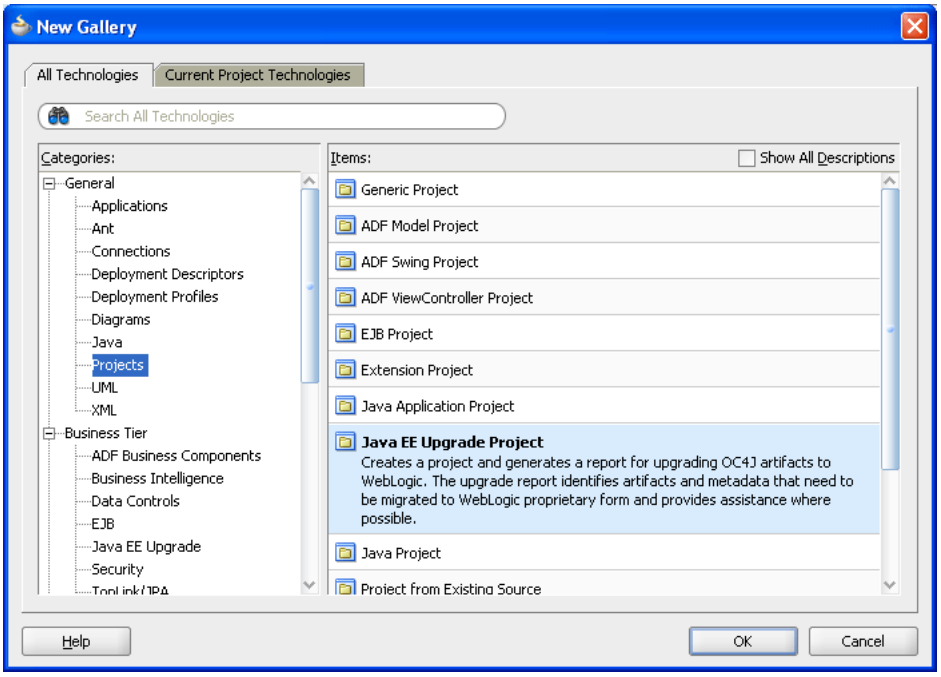
1. Choose **File** > **New** to display the New Gallery.

   You can also right-click an existing project and select **New** from the context menu.

2. In the Categories tree, select **General** then **Projects**.

3. In the Items list, double-click **Java EE Upgrade Project** (Figure 2–2).

Figure 2–2 Java EE Upgrade Project Item in the New Gallery Dialog



2.3.2 Using the Java EE Upgrade Wizard

Refer to the following sections for information about using the Java EE Upgrade wizard:

- Summary of the Java EE Upgrade Wizard Pages
- Types of Archives and Projects Supported by the SmartUpgrade Oracle JDeveloper Extension
- Limiting the Analysis to Specific SmartUpgrade Rule Categories
- Analyzing Applications Deployed on Oracle Application Server 10g Release 2 (10.1.2)

2.3.2.1 Summary of the Java EE Upgrade Wizard Pages

Table 2–1 describes the pages of the Java EE Upgrade wizard. Note that the pages that appear will vary depending on how you started the wizard and the type of report you want to generate.

Table 2–1 Summary of the Java EE Upgrade Application Wizard Pages

Page Name	Description	More Information
Application Name	This page appears only if you use the Java EE Upgrade Application wizard. Enter a name for the new Oracle JDeveloper application that SmartUpgrade creates after analyzing your OC4J application. The resulting upgrade report will be saved in a project inside the application.	Press F1 or click **Help** on the wizard page.
Project Name	Enter a name for the default project that SmartUpgrade creates after analyzing the selected application or project. The resulting upgrade report will be saved in this project.	Press F1 or click **Help** on the wizard page.
Upgrade Report Type (Figure 2–3)	Select the type of upgrade report you want to create. Note that if you are creating a new application, then the option to analyze an existing application does not appear on the page. To analyze one or more existing Oracle JDeveloper projects, you must use the Java EE Upgrade Project wizard.	Press F1 or click **Help** on the wizard page.
OC4J Artifacts (Figure 2–4)	The content of this page varies, depending upon the upgrade report type you select on the previous page. Select the enterprise archives, Oracle JDeveloper projects, or OC4J server directory you want to analyze, and specify any related options available on the page. In addition, you can: ■ Indicate whether or not you want to generate artifacts in addition to the upgrade report. ■ Click **Advanced** to limit the analysis to specific SmartUpgrade rules and to indicate the version of OC4J you are using.	Press F1 or click **Help** on the wizard page. Section 2.3.2.2, "Types of Archives and Projects Supported by the SmartUpgrade Oracle JDeveloper Extension" Section 2.3.2.3, "Limiting the Analysis to Specific SmartUpgrade Rule Categories" Section 2.3.2.4, "Analyzing Applications Deployed on Oracle Application Server 10g Release 2 (10.1.2)"
Required Inputs (Figure 2–5)	This page appears if the following are true: ■ You select **Generate Artifacts** on the OC4J Artifacts wizard page. ■ The application contains any Web services, data sources, EJBs, or other artifacts that can be upgraded by SmartUpgrade. ■ The rule categories selected in the Rules Management dialog include any of the object types for which SmartUpgrade can generate artifacts. You display the dialog box by clicking **Advanced** on the Required Inputs page.	Section 2.3.3, "Setting Properties on the Required Inputs Page" Section 2.3.2.2, "Types of Archives and Projects Supported by the SmartUpgrade Oracle JDeveloper Extension"

Figure 2–3 Java EE Upgrade Application Wizard - Upgrade Report Type Page

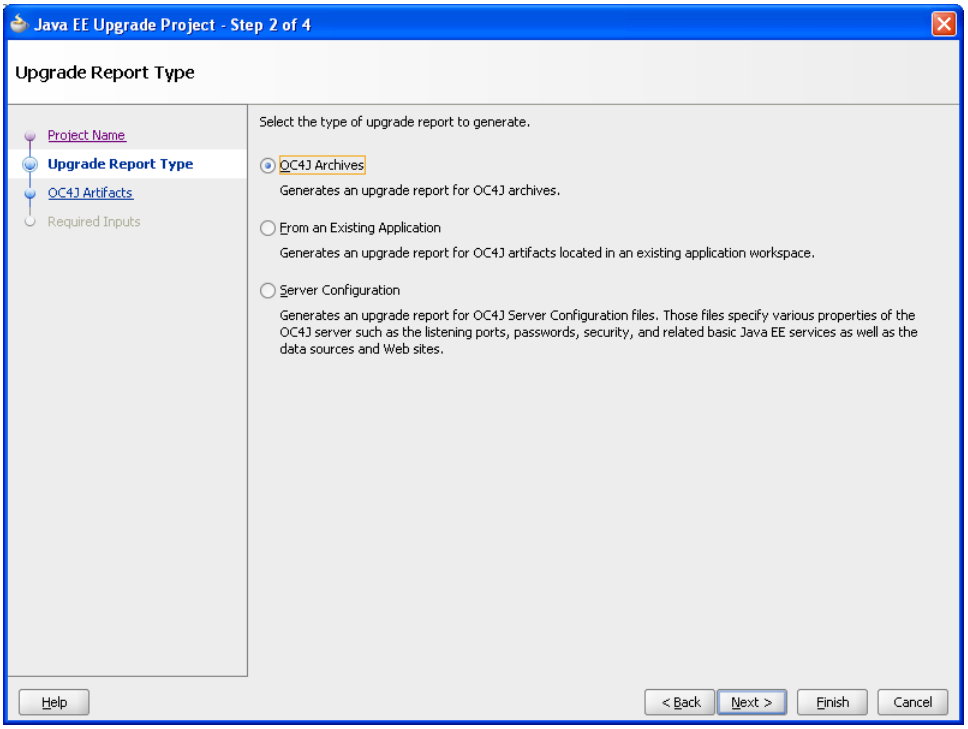


Figure 2–4 Java EE Upgrade Application Wizard - OC4J Artifacts Page When Analyzing an Enterprise Archive

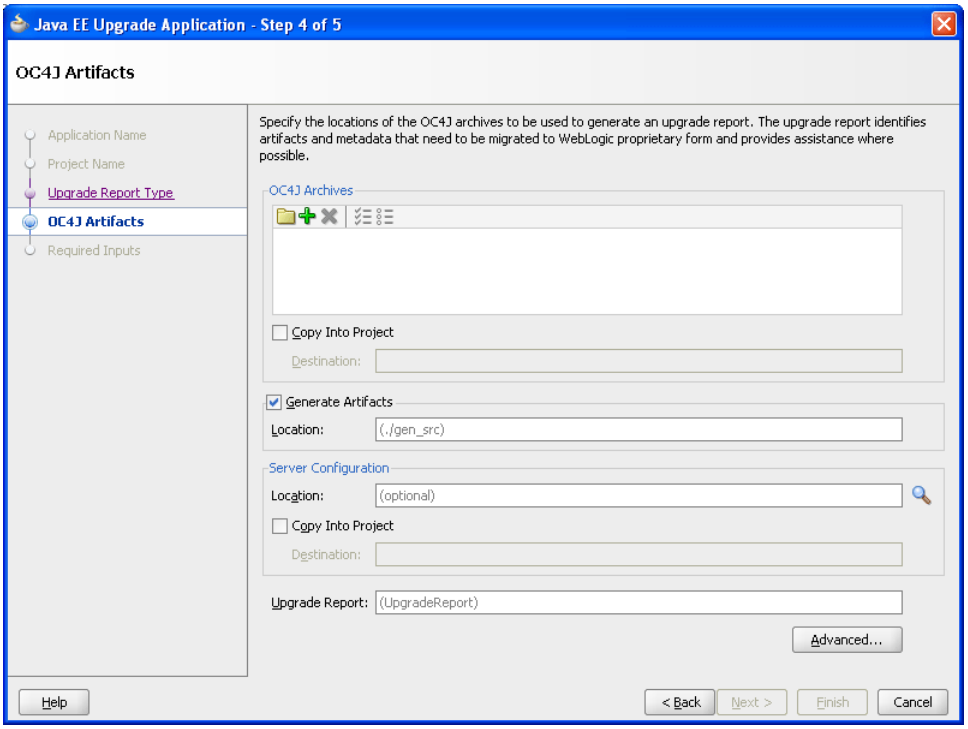
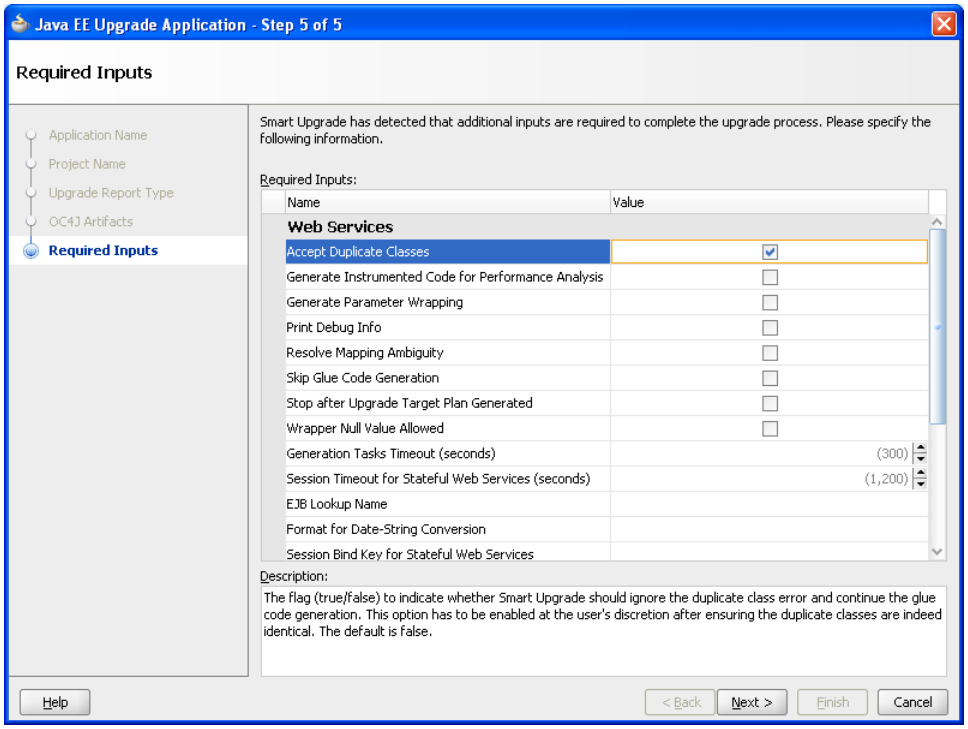


Figure 2–5 Java EE Upgrade Application Wizard - Required Inputs Page



2.3.2.2 Types of Archives and Projects Supported by the SmartUpgrade Oracle JDeveloper Extension

When running SmartUpgrade from Oracle JDeveloper, you can analyze the following:

- An existing enterprise archive (EAR) file, Web archive (WAR) file, Java archive (JAR) file, or WinRAR (RAR) file.
- One or more Oracle JDeveloper projects within the current Oracle JDeveloper application

If you want to analyze Web services within an application, you must select an EAR file. You cannot analyze or generate artifacts for Web services within an open Oracle JDeveloper application or project.

2.3.2.3 Limiting the Analysis to Specific SmartUpgrade Rule Categories

By default, SmartUpgrade generates a report and optionally generates artifacts by applying all rule categories to the selected archive or OC4J server configuration.

However, if you want to limit the size of the report, or if you want to focus on a particular aspect of your application, you can limit the analysis to a specific set of SmartUpgrade rule categories.

For example, to analyze only the data-source configuration of the `myApp.ear` application:

1. Start the Java EE Upgrade wizard, as described in Section 2.3.1, "Starting the Java EE Upgrade Wizard".
2. On the OC4J Artifacts page in the wizard, click **Advanced** to display the Rules Management dialog box.

3. Use the check boxes to select the rules categories you want SmartUpgrade to use during the current application analysis.

For a description of the SmartUpgrade rule categories you can apply when generating your reports and, optionally, your application artifacts, refer to Table 3–4.

2.3.2.4 Analyzing Applications Deployed on Oracle Application Server 10*g* Release 2 (10.1.2)

By default, SmartUpgrade is configured to analyze applications that were previously deployed on Oracle Application Server 10*g* Release 3 (10.1.3). However, if you are upgrading from 10*g* Release 2 (10.1.2), you can indicate the specific version of OC4J you are using:

1. Start the Java EE Upgrade wizard, as described in Section 2.3.1, "Starting the Java EE Upgrade Wizard".
2. On the OC4J Artifacts page in the wizard, click **Advanced** to display the Rules Management dialog box.
3. Select **10.1.3** or **10.1.2** from the OC4J Server Version section of the dialog box.

2.3.3 Setting Properties on the Required Inputs Page

If the application you are upgrading contains any Web services, data sources, EJBs or other object types for which SmartUpgrade can generate artifacts, then the Java EE Upgrade wizard usually displays the Required Inputs page (Figure 2–5).

For information about setting some the options on this page, refer to the following sections:

- Learning About Each of the Web Services Artifact Generation Options
- Generating Instrumented Code for Performance Analysis

2.3.3.1 Learning About Each of the Web Services Artifact Generation Options

You can learn more about the options on the page by referring to the following resources:

- Select a option on the page and review the text in the **Description** field.
- Table 3–2, " Summary of Optional SmartUpgrade Command-Line Options"
- Table 3–3, " Summary of the Command-Line Options Specific to Generating Artifacts"

2.3.3.2 Generating Instrumented Code for Performance Analysis

Select the **Generate Instrumented Code for Performance Analysis** property on the Required Inputs wizard page if you want to collect performance data and generate an HTML report that displays the performance impact of the additional code generated for the upgraded application.

After the upgraded Web service is deployed on Oracle WebLogic Server, it will generate an HTML report that summarizes the performance impact of the glue code. If there are multiple Web services, multiple report files will be generated.

For more information, see Section 4.2.3, "Analyzing the Performance Impact of the Web Services Glue Code Generated by SmartUpgrade".

2.3.4 Continuing with the Upgrade When Generating Artifacts

In some cases, when SmartUpgrade is analyzing the artifacts of your application, it might find that some manual steps are required before the application artifacts can be upgraded. In other cases, it might find that some properties that you did not provide are required.

In those cases, a finding appears in the Upgrade Report called, "Web Service Artifact Generation 10.1.3" or "Web Service Artifact Generation 10.1.2". This finding summarizes the state of upgrade process and provides the steps the user should take.

Note that a warning also appears in the SmartUpgrade log window at the bottom of the Oracle JDeveloper window.

In other cases, the "Web Service Artifact Generation" finding describes additional steps you must take before continuing with the Web services upgrade.

To run SmartUpgrade again and continue with the Web services upgrade process:

1. Right-click the upgrade report in the Oracle JDeveloper Application Navigator and select **Regenerate** from the context menu (Figure 2–6).

   Oracle JDeveloper displays the Report Generation dialog (Figure 2–7). Use this dialog to control whether the original findings in the report are overwritten on SmartUpgrade subsequent run.

2. If you are using the finding status to keep track of which findings have been implemented and which are currently in progress, select **All** or **Modified**.

   If you do not select **All** or **Modified**, then the entire report is regenerated and any status settings you selected for individual findings will be lost.

   For more information, see Section 2.4.5, "Using the Finding Status to Track Your Work".

3. Click **Next** to display the Required Inputs dialog box.

   Oracle JDeveloper displays the Required Inputs dialog, which is similar to the Required Inputs wizard page that appeared when you first ran SmartUpgrade, except that it contains only the properties you originally supplied, plus the additional properties that are required to continue the upgrade process.

4. Provide the missing property values and click **OK**.

   SmartUpgrade analyzes the Web services in the application again and attempts to complete the artifact generation. If additional properties are required, you might be prompted to continue the upgrade process again.

Figure 2–6 Selecting Regenerate from the Upgrade Report Context Menu

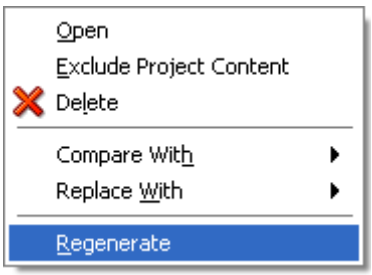
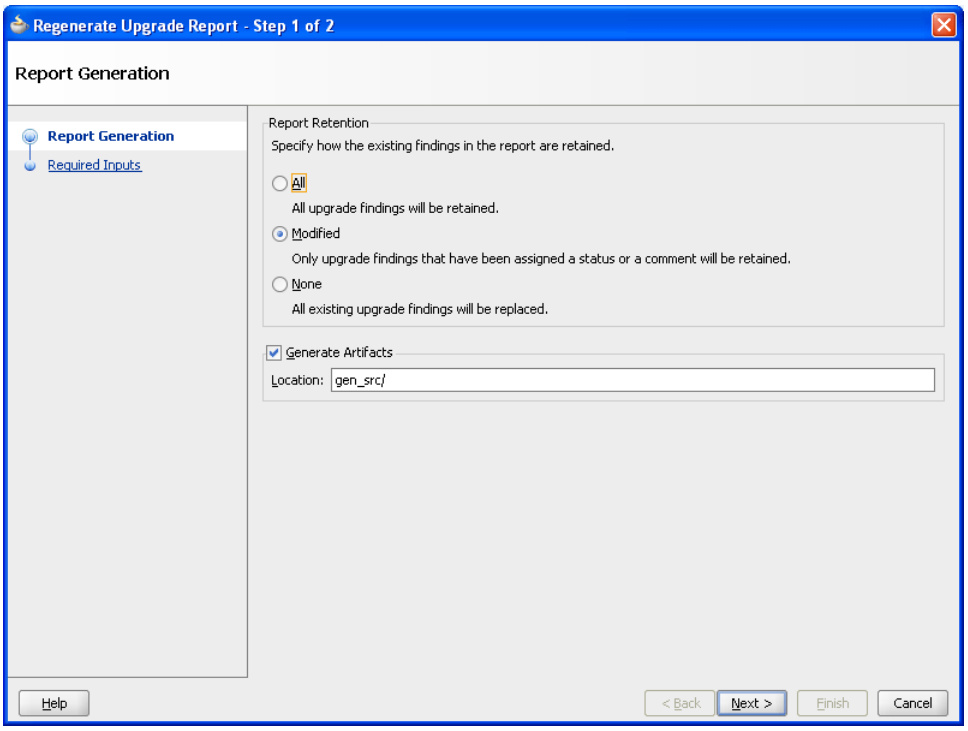


Figure 2–7 Report Generation Screen Displayed When You Choose to Regenerate the SmartUpgrade Report



2.4 Using a SmartUpgrade Report

After you generate a SmartUpgrade upgrade report in Oracle JDeveloper, you can sort and filter the findings. You can use this feature to focus on the most important findings first.

Refer to the following sections for more information:

- Viewing a SmartUpgrade Report
- Using the Upgrade Findings Toolbar to Filter SmartUpgrade Findings
- Using the Structure Window to Sort and Filter SmartUpgrade Findings

2.4.1 Viewing a SmartUpgrade Report

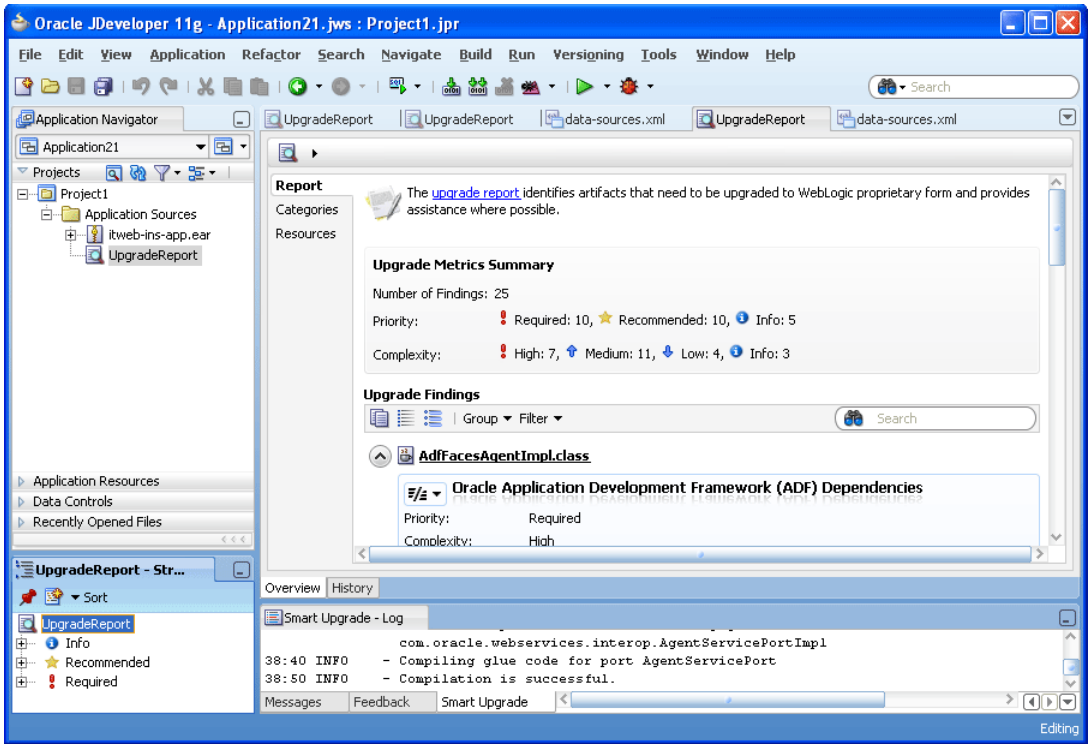
To view the upgrade report, use the Application Navigator to open the project you created with the Java EE Upgrade wizard. Locate the upgrade report under the Application Sources folder in the project and select the upgrade report (Figure 2–8).

Note that the report is divided into three tabs:

- The Report tab, which contains a summary of the report and a list containing each finding generated by SmartUpgrade for the selected application
- The Categories tab, which lists the categories of SmartUpgrade rules that were applied when generating the report.
- The Resources tab, which lists the archives or projects that were analyzed by SmartUpgrade to generate the report.

For more information about the types of information available for each finding on the Report tab, see Section 2.4.2, "Information Available in Each Finding".

Figure 2–8 SmartUpgrade Report in Oracle JDeveloper



2.4.2 Information Available in Each Finding

SmartUpgrade provides the following information within each finding within a SmartUpgrade report:

- A reason for the finding.

  The reason defines why the finding exists. It often identifies a particular element or attribute in the deployment descriptors of the application archive or in the configuration files of the OC4J server. It also identifies the dependency APIs used in the application, which need to be changed before the application can be deployed to Oracle WebLogic Server.

- Advice for how to remedy the problem.

  The advice identifies the actions you must take to resolve the issues identified in the finding. The advice often includes instructions or references to documentation that can help you modify the application or the Oracle WebLogic Server environment appropriately.

- The implications of the finding. The implication is optional (not always shown) and indicates the differences in behavior (if any) that will be noticed in the upgraded application as a result of following the finding's advice.

- Links to any generated artifacts associated with the finding. For example, for some data sources, SmartUpgrade generates samples XML code you can use to define the data sources in the application when it is deployed on Oracle WebLogic Server. For more information, see Chapter 4, "Using SmartUpgrade Generated Artifacts".

Figure 2–9 shows a sample finding within a SmartUpgrade report.

Figure 2–9 Sample Finding in a SmartUpgrade Report

2.4.3 Using the Upgrade Findings Toolbar to Filter SmartUpgrade Findings

If your report contains a long list of findings, you can easily group or filter the list using the Upgrading Findings toolbar (Figure 2–10).

Figure 2–10 Using the Upgrade Findings Toolbar

Table 2–2 describes each of the icons and menus available in the Upgrade Findings toolbar.

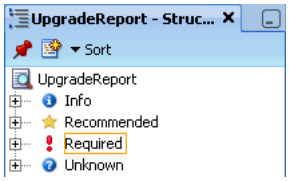
Table 2–2 Summary of the Upgrade Findings Toolbar Options

Icon or Menu	Description
	Click this icon to copy the contents of the report to the clipboard in regular text format. You can then paste it into a separate document or program.
	Click this icon to collapse all the findings so you can scroll down and see the main headings for each finding, but not the details.
	Click this icon to expand all the findings so the detail is displayed for every finding in the report.
Group	Use this submenu to group the findings by finding by artifact, category, complexity, or priority. For example, if you group the findings by priority, all the required findings are listed in the report first, followed by all the findings that SmartUpgrade identifies as recommended, followed by all the findings that are informational. You can also control whether or not the groups are sorted in ascending or descending order. For example, if group your findings by category, you can determine which category of findings appears at the top of the report.
Filter	Use this submenu to filter the findings by type, priority, or category. When you select an item from this menu, all the findings that do not meet the selected criteria are hidden. For example, you can filter out all but the required findings.

2.4.4 Using the Structure Window to Sort and Filter SmartUpgrade Findings

The structure window (Figure 2–11) under the Oracle JDeveloper navigation pane provides a view of the report structure. You can use this structure window to filter the findings in your report.

Figure 2–11 SmartUpgrade Report Structure Window



In this default view, you can quickly view all findings of a particular priority. For example, click **Required** to show only the findings that are marked as "Required."

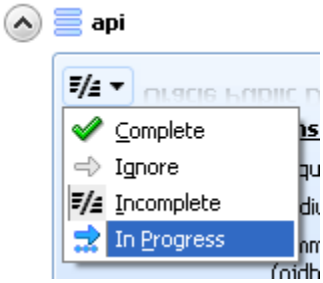
You can also use the **Sort** menu in the Structure view to change the list. For example, select **Complexity** from the **Sort** menu to show a list of the complexity categories. You can then select one of the categories to view only the findings of that category in the Report window.

At any time, you can click the root node in the structure view (UpgradeReport in Figure 2–11) to view all the findings in the report.

2.4.5 Using the Finding Status to Track Your Work

As you review the Upgrade report, note that you can track your progress using the Finding Status menu (Figure 2–12).

Figure 2–12 The Finding Status Menu



For example, for each finding that you finish implementing, select **Complete**. For those that are not implemented, use the **Ignore**, **Incomplete**, or **In Progress** settings. You can then quickly scan the report to see how much remaining work is required.

Note that if you need to regenerate the report, you must select All or Modified on the Report Generation dialog if you want to don't want the new report to overwrite your status settings.

For more information, see Section 2.3.4, "Continuing with the Upgrade When Generating Artifacts".

3

Using the SmartUpgrade Command Line

In addition to using SmartUpgrade as an integrated Oracle JDeveloper extension, you can also use the SmartUpgrade command-line interface.

The command line interface can be run using the Java command line or as an Apache Ant task.

With the SmartUpgrade command-line, you can consider automating the analysis and generation of artifacts for multiple applications using scripts.

For more information, see the following sections:

- Using the SmartUpgrade Java Command-Line
- Integrating SmartUpgrade with Apache Ant

3.1 Using the SmartUpgrade Java Command-Line

The following sections describe how to use the command-line interface for SmartUpgrade:

- Verifying Prerequisites and Locating the smartupgrade.jar File
- Basic Syntax of the SmartUpgrade Command-Line Interface
- Getting Help on the SmartUpgrade Command-Line Options
- Summary of the SmartUpgrade Command-Line Options
- Summary of the SmartUpgrade Command-Line Options Specific to Artifact Generation
- Examples of Using the SmartUpgrade Command-Line Interface

3.1.1 Verifying Prerequisites and Locating the smartupgrade.jar File

Before you can run the SmartUpgrade command-line interface:

1. Verify that you have installed the required Java software, as described in Section 1.6.3, "Installing the SmartUpgrade Command-Line Interface".

   Note that you should also make sure that the Java bin directory is defined as part of the current PATH variable, so you can run the `java` command from any location on your system. Otherwise, you must include the path to the `java` command every time you run the SmartUpgrade software.

2. Verify that you have installed a supported version of Oracle WebLogic Server and that you have access to a local Oracle WebLogic Server home directory.

You specify the location of the Oracle WebLogic Server home directory using the `-targetStackHome` command-line option.

For more information, see Table 3–3, " Summary of the Command-Line Options Specific to Generating Artifacts".

3. Locate the `smartupgrade.jar` file, which you installed using the instructions in Section 1.6, "Downloading and Installing SmartUpgrade".

Note that the instructions and examples in this chapter assume you are running SmartUpgrade from the directory where the `smartupgrade.jar` resides.

3.1.2 Basic Syntax of the SmartUpgrade Command-Line Interface

To use the SmartUpgrade command-line interface, simply navigate to the directory where you unpacked the contents of the downloaded smartupgrade.zip file and use the following syntax:

```
java -jar smartupgrade.jar options
```

For more information, see Section 3.1.5, "Summary of the SmartUpgrade Command-Line Options".

3.1.3 Optionally Increasing the Java Heap Size When Running SmartUpgrade

As with other Java programs, you can control the Java heap size when you run SmartUpgrade.

Increasing the heap size can be useful if you are upgrading a particularly large or complex application. In those cases, SmartUpgrade displays a message similar to the following:

```
java.lang.OutOfMemoryError: Java heap space
```

To work around this problem, increase the Java heap size using the standard Java command line. For example:

```
java -Xmx512M -jar smartupgrade.jar options
```

3.1.4 Getting Help on the SmartUpgrade Command-Line Options

To display a list of the available options, enter one of the following commands:

```
java -jar smartupgrade.jar -help
java -jar smartupgrade.jar -help locator
java -jar smartupgrade.jar -help category
java -jar smartupgrade.jar -help option
```

For detailed information, see Section 3.1.5, "Summary of the SmartUpgrade Command-Line Options".

3.1.5 Summary of the SmartUpgrade Command-Line Options

Refer to the following sections for detailed information about the options you can use when running SmartUpgrade from the command line:

- List of SmartUpgrade Command-Line Interface Options
- Identifying a SmartUpgrade Locator
- Specifying More Than One Locator on the SmartUpgrade Command Line

- Summary of the SmartUpgrade Optional Command-Line Options

3.1.5.1 List of SmartUpgrade Command-Line Interface Options

The following example shows the options you can use when running the SmartUpgrade command-line utility:

```
java -jar smartupgrade.jar --LOCATOR_NAME path_or_list_of_file_names
                           -category list_of_categories
                           -generate
                           -html
                           -target OC4J_version
```

In the previous example:

- Replace LOCATOR_NAME with a valid Locator that SmartUpgrade upgrade can analyze. For more information, see Section 3.1.5.2, "Identifying a SmartUpgrade Locator".
- Note that the two dashes (`--`) are required to identify the LOCATOR_NAME. The LOCATOR_NAME must be prefixed with the two dashes. All other arguments require only one dash.
- All but the LOCATOR_NAME value are optional.
- For detailed information about the other options shown in the example, see Table 3–2.

3.1.5.2 Identifying a SmartUpgrade Locator

A locator is a general term to identify the object or objects that you want SmartUpgrade to analyze. The locator can be one or more application archives (EAR, WAR, JAR, or RAR files). It can also be a directory path where archives are stored, or the configuration directory of an OC4J server.

Table 3–1 describes the values you can use for the LOCATOR_NAME command-line option.

Note that if you use a LOCATOR_NAME option that does not match the type of file or configuration you want to upgrade, SmartUpgrade can produce incomplete or invalid output. For example, the SmartUpgrade output might be incomplete or invalid output if you use the `--ears` LOCATOR_NAME value and provide the name of a JAR file, rather than an EAR file.

Table 3–1 Supported Values for the SmartUpgrade LOCATOR_NAME Option

***LOCATOR_NAME* value**	**Description**	**Examples**

--ears	Identifies one or more enterprise archive (EAR) files to analyze.
	If you are a providing the path to more than one EAR file, then use a space-delimited list after the -ear option.
		`java -jar smartupgrade.jar --ears myApp.ear`
		`java -jar smartupgrade.jar --ears C:\samples\App3.ear`
		`java -jar smartupgrade.jar --ears myApp.ear App3.ear`

--wars	Identifies one or more Web archive (WAR) files to analyze.
	If you are a providing the path to more than one WAR file, then use a space-delimited list after the -ear option.
		`java -jar smartupgrade.jar --wars payroll.war`
		`java -jar smartupgrade.jar --wars C:\samples\Webapp3.war`
		`java -jar smartupgrade.jar --wars payroll.war C:\samples\Webapp3.war`

Table 3–1 (Cont.) Supported Values for the SmartUpgrade LOCATOR_NAME Option

***LOCATOR_NAME* value**	**Description**	**Examples**

--jars	Identifies one or more Java archive (JAR) files to analyze.
	If you are a providing the path to more than one JAR file, then use a space-delimited list after the -ear option.
		`java -jar smartupgrade.jar --jars myProj.jar`
		`java -jar smartupgrade.jar --jars C:\samples\App3.jar`
		`java -jar smartupgrade.jar --jars myApp.jar C:\samples\App3.jar`

--rars	Identifies one or more RAR archive files to analyze.
	If you are a providing the path to more than one RAR file, then use a space-delimited list after the -ear option.
		`java -jar smartupgrade.jar --rars myApp.rar`
		`java -jar smartupgrade.jar --rars C:\samples\App3.rar`
		`java -jar smartupgrade.jar --rars myApp.rar C:\samples\App3.rar`

--server-config	Identifies the configuration directory of an existing OC4J server. SmartUpgrade analyzes the configuration of the OC4J server and provide advice and configuring Oracle WebLogic Server in a similar manner.	`java -jar smartupgrade.jar --server-config C:\Oracle\AppServ1\j2ee\home\config` `java -jar smartupgrade.jar --server-config /dua1/Oracle/AppServ1/j2ee/home/config`
--archive-homes	Identifies one or more directories that contain archive files (EAR, WAR, RAR, or JAR files) that you want to analyze. SmartUpgrade scans the directory and analyzes all the archives in the directory.	`java -jar smartupgrade.jar --archive-home C:\projects\myEARfiles\` `java -jar smartupgrade.jar --archive-home /dua1/projects/myEARfiles/`
--application-jars	Identifies the location of jar files that required or referenced by an application you are analyzing. The specified file is not analyzed by SmartUpgrade. You can use this feature to identify third-party libraries required by the application you are analyzing.	`java -jar smartupgrade.jar --application-jars C:\projects\myApp\lib\` `java -jar smartupgrade.jar --application-jars /dua1/projects/myApp/lib/`

3.1.5.3 Specifying More Than One Locator on the SmartUpgrade Command Line

You can specify more than one LOCATOR_NAME on the command line.

For example, to analyze an enterprise archive and the configuration of the OC4J server on Linux where the archive was deployed, use the following command:

```
java -jar smartupgrade.jar --ears myApp.ear
                           --server-config /dua1/Oracle/AppServ1/j2ee/home/config
```

3.1.5.4 Summary of the SmartUpgrade Optional Command-Line Options

In addition to identifying a SmartUpgrade locator, you can also control the behavior of SmartUpgrade by using various optional command-line options.

Table 3–2 describes the optional command-line options you can use.

Table 3–2 Summary of Optional SmartUpgrade Command-Line Options

Options	Description	For More Information
-category	Use this option to limit the analysis of the application to specific categories of SmartUpgrade rules. For example, you can limit the report to findings about data-source configurations in the selected application archive. If you are generating artifacts, use caution when using the `-category` option. If you use both the `-category` and `-generate` options, SmartUpgrade generates only the artifacts of the specified categories.	Section 3.1.7.5, "Limiting the Findings to Specific Rule Categories from the SmartUpgrade Command-Line Interface"
-generate	Use this option to generate specific types of Oracle WebLogic Server artifacts, such as Oracle WebLogic Server deployment descriptors for the application. The artifacts can be used as a starting point for the deploying your OC4J applications on Oracle WebLogic Server.	Section 3.1.7.3, "Using the SmartUpgrade Command-Line Interface to Generate Oracle WebLogic Server Artifacts"
-html	Use this option to generate output formatted in HTML. You can use this option to redirect the resulting report to an HTML file.	Section 3.1.7.4, "Using the SmartUpgrade Command-Line Interface to Generate an HTML Report"
-target	Use this option to specify that you are analyzing an Oracle Application Server 10*g* Release 2 (10.1.2) application. By default, SmartUpgrade assumes you are analyzing an application that was previously deployed on 10*g* Release 3 (10.1.3)	Section 3.1.7.6, "Using the SmartUpgrade Command-Line Interface to Analyze 10g Release 2 (10.1.2) Applications"
-output	Use this option to output all the upgrade findings to file.	Section 3.1.8, "Controlling the Output of SmartUpgrade Findings Information and Error Reporting" Section 3.1.7.4, "Using the SmartUpgrade Command-Line Interface to Generate an HTML Report"
-quiet	Use this option to prevent SmartUpgrade from outputting any error or diagnostic information.	Section 3.1.8, "Controlling the Output of SmartUpgrade Findings Information and Error Reporting"

3.1.6 Summary of the SmartUpgrade Command-Line Options Specific to Artifact Generation

Example 3–1 shows the options you can use when running the SmartUpgrade command-line utility to generate artifacts.

For complete information about using the artifacts generated by SmartUpgrade, refer to Chapter 4, "Using SmartUpgrade Generated Artifacts".

For a description of each option, refer to Table 3–3, " Summary of the Command-Line Options Specific to Generating Artifacts".

When generating Web services artifacts, the LOCATOR_NAME value, `-generate` option, and `-targetStackHome` option are mandatory.

Replace LOCATOR_NAME with a valid Locator that SmartUpgrade upgrade can analyze. For more information, see Section 3.1.5.2, "Identifying a SmartUpgrade Locator".

Example 3–1 List of Command-Line Options When Generating Artifacts

```
java -jar smartupgrade.jar --LOCATOR_NAME path_to_application_archive_or_directory
          -generate
          -category category_name
                      -acceptDuplicates
                      -additionalClassPath
                      -autoWrap
                      -continue
                      -dateFormat
                      -debug
                      -ejbLookupName
                      -ejbNewWarBase
                      -ejbNewWarContextRoot
                      -evaluate
                      -javaHome
                      -jdevProject
                      -logLevel
                      -logWrap
                      -logWrapLength
                      -packLibs
                      -processTimeout
                      -propertyFile
                      -qos
                      -resolveMapAmbiguity
                      -sessionImplKey
                      -sessionTimeoutSecs
                      -skipGlueCode
                      -stopAtTargetPlan
                      -targetStackHome
                      -useJSF
                      -useJSTL
                      -wrapperNullAllowed
```

Table 3–3 Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-acceptDuplicates`	Accept Duplicate Classes	Enable this option by passing `-acceptDuplicates` (or `-acceptDuplicates true`) on the command line or by selecting the check box on the OC4J Web Services Upgrade wizard page. This option indicates whether the system should ignore the duplicate class error and continue the glue code generation. You should enable this option only after ensuring that the duplicate classes are identical.
`-additionalClassPath`	Additional Classpath	An additional classpath containing JAR files and class directories. This option is required when the input application does not contain all the libraries that a Web service depends on. This option can also be used to provide a path to a missing WSDL file, inserted in a JAR file.

Table 3–3 (Cont.) Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-autoWrap`	Generate Parameter Wrapping	Enable this option by passing `-autoWrap` (or `-autoWrap true`) on command line or by selecting the check box on the OC4J Web Services Upgrade wizard page. When this option is enabled, SmartUpgrade enables parameter wrapping for the internal types that are generated. The existing business methods will not be aware of any of new interfaces and value types generated. When this option is not specified, its value is defined by whether the existing business parameters are wrapped.
`-continue`	1. Select **Regenerate** from the SmartUpgrade Report context menu. 2. Select **Continue with web service upgrade** on the Required Inputs screen.	This option applies specifically to artifact generation of Web Services. It should be used in conjunction with '-generate' argument when category 'web-services' is enabled. This option indicates that the web service upgrade should proceed with the upgrade generation process which was started in a previous invocation of SmartUpgrade. For more information, see Section 2.3.4, "Continuing with the Upgrade When Generating Artifacts".
`-dateFormat`	Format for Date-String Conversion	The date format to be used by glue code for conversion between `java.lang.String` and `java.util.Date`. If SmartUpgrade detects that WebLogic Web Services generation tools have generated `java.util.Date` value types, but the original application methods use `java.lang.String`, then SmartUpgrade will add conversion methods to the glue code. However, SmartUpgrade will need the string format for this conversion. As a result, you must examine the OC4J application to determine the specific string format that the existing business logic expects for its date values.
`-debug`	Print Debug Info	Enable this option by passing `-debug` (or `-debug true`) on the command line or by selecting the check box on the OC4J Web Services Upgrade wizard page. When enabled, indicates that debug level messages should be printed. The debug messages are more detailed than INFO level messages. The default value is false.
`-ejbLookupName`	EJB Lookup Name	The EJB 2.x lookup name to be used in the POJO-based Web service to which an EJB web service will be upgraded. This option should be used only when there is a single EJB Web service in the application. If multiple EJB Web services are present, edit the upgrade plans generated for each service. For information about locating the upgrade plans, see Appendix A, "Output Directories Generated by SmartUpgrade".

Table 3–3 (Cont.) Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-ejbNewWarBase`	WAR Base Name	The base name of the WAR file to be generated. When an EJB-based Web service is upgraded to a POJO-based Web service, SmartUpgrade generates a new WAR file. The value of this option will be the base name of the new WAR file. If there are more than one EJB Web service, edit the source upgrade plan to add this property with different values for each EJB Web service. For information about locating the upgrade plans, see Appendix A, "Output Directories Generated by SmartUpgrade".
`-ejbNewWarContext`	WAR Context Root	The context path of the WAR file to be generated. When an EJB-based Web service is upgraded to POJO-based Web service, SmartUpgrade generates a new WAR file. The value of this option will be the context root of the new WAR file. If there is more than one EJB Web service, then modify the source upgrade plan to add this property with different values for each EJB Web service. For information about locating the upgrade plans, see Appendix A, "Output Directories Generated by SmartUpgrade".
`-evaluate`	Generate Instrumented Code for Performance Analysis	Enable this option by passing `-evaluate` (or `-evaluate true`) on command line, or by selecting the check box on the OC4J Web Services Upgrade wizard page. When you enable this option, the generated wrapper ("glue") code is instrumented for performance analysis. The default value of the option is `false`. This option is appropriate only for testing the application, and not for production environments. For more information, see Section 2.3.3.2, "Generating Instrumented Code for Performance Analysis".
`-javaHome`	JDK Home for Target Server Tools	The JDK home that SmartUpgrade uses to run the Web Service tools on the target server. The system automatically attempts to locate JDK installed with the target server. You can override this behavior by specifying a JDK home as a value for this option. If this option is not specified and if no JDK home is installed as part of the target server, the system will use the current JDK home used to run this upgrade tool.
`-jdevProject`	Not applicable	This is an option used for Oracle internal use only.

Table 3–3 (Cont.) Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-logLevel`	Logging Level	Use this option to set the level of logging output. This option indicates the levels of logging output during upgrade of Web services. Possible log levels are: INFO, FINE, DEBUG, TRACE. The default value is INFO. Warnings and errors are reported unconditionally. Every log message belongs to one log level and will be printed only if its log level is requested. Specify all levels that you wish to be included in the log, separated by the vertical bar character (|). For example: INFO|DEBUG. **Note:** This option is not validated; any invalid value will be ignored.
`-logWrap`	Margins in Log Output	Enable this option by passing `-debugWrap` (or `-debugWrap true`) on the command line or by selecting the check box on the OC4J Web Services Upgrade wizard page. When enabled, this option causes SmartUpgrade to align the diagnostic output for easy reading. The default value is `true`. You can adjust column width using the `logWrapLength` option.
`-logWrapLength`	Log Line Width	Use this option to set the maximum length of the text that is logged without breaking the text into multiple lines. Use this option to preserve alignment. This is applicable only when `logWrap` option is set to `true`. The default value is `80`.
`-packLibs`	Packaging Includes	A list of JAR files, separated by a path separator. On Windows, the separator is a semicolon (;); on UNIX systems, the separator is colon (:). The identified JAR files are automatically packaged into the final generated archive, if the final archive is an EAR or WAR file. This value is also added into the value of `additionalClassPath` option to use the libraries for reference during artifact generation.
`-processTimeout`	Generate Tasks Timeout (seconds)	The number of seconds SmartUpgrade will continue trying to process WSDL documents and generate artifacts. The timeout prevents SmartUpgrade from hanging when waiting for response from a WSDL URL. The tasks timeout default is 300 seconds. A different value may be set by using this command-line option, by providing a positive number on the wizard page in Oracle JDeveloper, or by setting the following environment variable prior to starting SmartUpgrade: GENERATION_TASKS_TIMEOUT

Table 3–3 (Cont.) Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-propertyFile`	Property File	This option applies specifically to artifact generation of Web Services. It should be used in conjunction with '-generate' argument when category 'web-services' is enabled. The location of a property file where the users usually define the location of missing WSDLs with the key being the SEI class name and the value being the path of the WSDL file. Other properties may be defined in this file instead of using separate options, such as Format for Date-String Conversion, Wrapper Null Value Allowed, EJB Lookup Name, WAR Base Name for EJB Web Service, WAR Context Root for EJB Web Service, Session Bind Key for Stateful Web Services and SessionTimeout for Stateful Web Services (seconds).
`-qos`	Quality of Service Policies	This option indicates the types of policies (Quality Of Services) that need to be enabled in the target Web service that is generated. The list of possible tokens is MTOM, WSS_UNT, CONVERSATIONAL, STATEFUL, or RM11. Multiple values can be passed with a comma(,) as the separator.
`-resolveMapAmbiguity`	Resolve Mapping Ambiguity	Enable this option by setting the value to `true` or selecting the check box in the OC4J Web Services Upgrade wizard page. Enable this option to allow the system to resolve mapping ambiguities automatically. This option is applicable only when SmartUpgrade does not find mapping information either in mapping file or from annotations. The default value is `false`. If you want to rely on SmartUpgrade to resolve ambiguity, then enable this option.
`-sessionImplKey`	Session Bind Key for Stateful Web Services	This option applies specifically to artifact generation of Web Services. It should be used in conjunction with '-generate' argument when category 'web-services' is enabled. The bind key used to bind the business implementation to the HTTP session in the case of a Stateful web service. The default value is derived from the generated implementation class. The default value can be overridden at the web service level.
`-sessionTimeoutSecs`	Session Timeout for Stateful Web Services (seconds)	This option applies specifically to artifact generation of Web Services. It should be used in conjunction with '-generate' argument when category 'web-services' is enabled. The web service session timeout in seconds. This value can not be more than the HTTP session timeout of the application. The default value is 1,200.

Table 3–3 (Cont.) Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-skipGlueCode`	Skip Glue Code Generation	Enable this option by setting the value to `true` or selecting the check box in the OC4J Web Services Upgrade wizard page. When enabled, this option indicates that glue code generation needs to be skipped. When enabled, the target plan generation is also skipped. Use this option when the you have modified the generated glue code and you need to compile and package the application.
`-stopAtTargetPlan`	Stop after Upgrade Target Plan Generated	This option applies specifically to artifact generation of Web Services. It should be used in conjunction with '-generate' argument when category 'web-services' is enabled. The flag (true/false) to indicate whether the glue code generation should be done. The default value is false - glue code generation will be done. When option value is true, web services upgrade stops once the target upgrade plan is generated. The source upgrade plan, unless skipped, will be generated before generating target upgrade plan. The option is used when the users want to edit the target upgrade plan and then generate glue code in a separate step.
`-targetStackHome`	Not Applicable	Mandatory command-line property used to specify the target WebLogic Server home. For example: `C:\Oracle\Middleware\wlserver_10.3` If you are migrating a large number of applications, you can avoid repeatedly specifying this property by setting the environment variable WL_HOME prior to starting Oracle JDeveloper. The value you set in WL_HOME environment variable will be applied to all applications during the upgrade process. Note that this property is not necessary when using Oracle JDeveloper, because the SmartUpgrade Oracle JDeveloper extension automatically uses the libraries and other required files from the Oracle WebLogic Server environment installed with Oracle JDeveloper. For more information, see Section 3.1.7.1, "Identifying the Oracle WebLogic Server Home Directory".
`-useJSF`	Use latest JSF Libraries	The JSF libraries are provided as Web Application libraries and must be deployed to WLS for this Web Application to utilize JSF features.
`-useJSTL`	Use latest JSTL Libraries	The JSTL libraries are provided as Web Application libraries and must be deployed to WLS for this Web Application to utilize JSTL features.

Table 3–3 (Cont.) Summary of the Command-Line Options Specific to Generating Artifacts

Command-Line Option	Equivalent Option in the Java EE Upgrade Wizard	Description
`-wrapperNullAllowed`	Wrapper Null Value Allowed	This option applies specifically to artifact generation of Web Services. It should be used in conjunction with the `-generate` argument when the `web-services` category is enabled. Enable this option by passing `-wrapperNullAllowed` (or `-wrapperNullAllowed true`) on the command line or by selecting the check box on the OC4J Web Services Upgrade wizard page. When enabled, null values are allowed for wrapper types already present in the application before upgrade. The default value is `false`. In this case, the null value of a wrapper type will be represented by an empty object. Note that WebLogic Server JAX-RPC Web Services do not allow null values for wrapper types.

3.1.7 Examples of Using the SmartUpgrade Command-Line Interface

The following sections provide some examples of how you can use the SmartUpgrade command-line options to help you upgrade your applications to Oracle WebLogic Server:

- Identifying the Oracle WebLogic Server Home Directory
- Using the SmartUpgrade Command-Line Interface to analyze an Enterprise Archive (EAR) File
- Using the SmartUpgrade Command-Line Interface to Generate Oracle WebLogic Server Artifacts
- Using the SmartUpgrade Command-Line Interface to Generate an HTML Report
- Limiting the Findings to Specific Rule Categories from the SmartUpgrade Command-Line Interface
- Using the SmartUpgrade Command-Line Interface to Analyze 10g Release 2 (10.1.2) Applications

3.1.7.1 Identifying the Oracle WebLogic Server Home Directory

For many of its operations, SmartUpgrade requires specific libraries and resources that are provided by Oracle WebLogic Server.

When running SmartUpgrade as an extension to Oracle JDeveloper, SmartUpgrade locates these files automatically using the Oracle WebLogic Server software that you install with the Oracle JDeveloper installer. For more information, see Section 2.1, "Installing and Configuring Oracle JDeveloper".

However, on the command line, you must either define the environment variable WL_HOME or use the `-targetStackHome` option to specify the location of an Oracle WebLogic Server home directory to use for this purpose.

For example, to define the WL_HOME variable:

```
set WL_HOME=c:\middleware\wlserver_10.3
```

To use the `-targetStackHome` option, refer to the following example:

```
java -jar smartupgrade.jar
     --ears myApp.ear
     -generate
     -category web-services
     -targetStackHome c:\middleware\wlserver_10.3
```

3.1.7.2 Using the SmartUpgrade Command-Line Interface to analyze an Enterprise Archive (EAR) File

To generate a SmartUpgrade report for a specific enterprise archive (EAR) file, use the following command syntax:

```
java -jar smartupgrade.jar --ears path_of_ear_file.ear
```

For example, if you have an EAR file called `myApp.ear` and it resides in a directory called my `C:\MyApps`, then use the following command:

```
java -jar smartupgrade.jar --ears C:\MyApps\myApp.ear
```

SmartUpgrade generates a report that is written to the terminal window. To save the report findings, redirect the output to a file. For example, the following examples show how to redirect the output to a file called `report.txt`.

- On the Windows operating system:

  ```
  java -jar smartupgrade.jar --ears C:\MyApps\myApp.ear > account_report.txt
  ```

- On the UNIX operating system:

  ```
  java -jar smartupgrade.jar --ears /home/MyApps/myApp.ear > account_report.txt
  ```

3.1.7.3 Using the SmartUpgrade Command-Line Interface to Generate Oracle WebLogic Server Artifacts

In addition to generating a report, SmartUpgrade can also generate a limited number of artifacts that can make it easier to update your applications so they can be deployed successfully on Oracle WebLogic Server.

At a minimum, you must:

- Use the `-generate` option to generate artifacts
- Specify the mandatory `-targetStackHome` option that identifies the target server home directory

For example, to generate artifacts in addition to a SmartUpgrade report for an archive called `myApp.ear`, use the following command:

```
java -jar smartupgrade.jar
     --ears archive_name
     -generate
     -targetStackHome c:\middleware\wlserver_10.3
```

For another example, suppose you want to do the following:

- Specify the Oracle WebLogic Server home
- Restrict generation of artifacts for upgrade to Web Services artifacts only
- Specify the classpath to additional libraries not contained in the application archive but required for the loading of classes
- Indicate that you want SmartUpgrade to generate code instrumented for performance analysis

In that case, you would enter the following command:

```
java -jar smartupgrade.jar
     --ears archive_name
      -generate
      -category web-services
               -targetStackHome c:\middleware\wlserver_10.3
               -additionalClassPath path_to_libraries
               -evaluate
```

For information on the results of this analysis, see Chapter 4, "Using SmartUpgrade Generated Artifacts".

3.1.7.4 Using the SmartUpgrade Command-Line Interface to Generate an HTML Report

By default, the SmartUpgrade command-line interface generates a report in text format and the report is written to the standard output of the current machine. In most cases, this means the output is written to the terminal window.

You can optionally generate a report in HTML format, which includes headings and lists that can make the output easier to read. Further, you can use your operating system commands to redirect the output to a file. The resulting HTML file can be read by any Web browser or other tool that can read HTML.

For example, if you have an EAR file called `account_mgmt.ear` and it resides in a directory called my `C:\MyApps`, then use the following command to generate an HTML file called `account_mgmt_report.html`:

```
java -jar smartupgrade.jar --ears myApp.ear -html > account_mgmt_report.html
```

For related information, see Section 3.1.8, "Controlling the Output of SmartUpgrade Findings Information and Error Reporting".

3.1.7.5 Limiting the Findings to Specific Rule Categories from the SmartUpgrade Command-Line Interface

By default, SmartUpgrade generates a report and optionally generates artifacts by applying all rule categories to the selected archive or OC4J server configuration.

However, if you want to limit the size of the report, or if you want to focus on a particular aspect of your application, you can limit the analysis to a specific set of SmartUpgrade rule categories.

For example, to analyze only the data-source configuration of the `myApp.ear` application:

```
java -jar smartupgrade.jar --ears myApp.ear -category data-sources
```

To apply multiple categories, separate the list of categories with a space. For example:

```
java -jar smartupgrade.jar --ears myApp.ear -category data-sources web-app
```

Table 3–4 describes the SmartUpgrade rule categories you can apply when generating your reports and, optionally, your application artifacts.

> **Note:** If you are generating artifacts, use caution when specifying the `-category` option. If you use both the `-category` and `-generate` options, SmartUpgrade generates only the artifacts of the specified categories.

Table 3–4 List of SmartUpgrade Rule Categories

Rule Category	Use this category to analyze...
adf	Artifacts specific to the Oracle Application Development Framework (Oracle ADF).
api	Standard application programming interfaces (APIs) used in the application. This category of rules checks for OC4J and third-party APIs that may not be supported in Oracle WebLogic Server.
app-client	Client interfaces within the application.
classloading	Classloading configurations and shared libraries used by the application.
cluster	Cluster-specific configuration settings in the application.
data-sources	OC4J-specific data source configuration settings.
ejb	Enterprise Java Beans being used by the application.
jca	J2EE Connector Architecture (JCA) configuration settings and artifacts.
jms	Java Messaging Server configuration settings and artifacts, including the use of Oracle Enterprise Messaging Service (OEMS).
jmx	Java Management Extensions (JMX) used by the application.
jndi	Java Naming and Directory Interface (JNDI) configuration settings and artifacts.
jta	Java Transaction API (JTA) configuration settings and artifacts.
rmi	Remote Method Invocation (RMI) configuration settings and artifacts.
security	Security configuration settings and artifacts.
soa	Service-Oriented Architecture (SOA) configuration settings and artifacts.
web-app	Web application configuration settings and artifacts.
web-services	Web services configuration settings and artifacts.
webcache	Oracle Web Cache configuration settings.

3.1.7.6 Using the SmartUpgrade Command-Line Interface to Analyze 10*g* Release 2 (10.1.2) Applications

By default, SmartUpgrade assumes the applications you are upgrading were previously deployed on Oracle Application Server 10*g* Release 3 (10.1.3).

However, you can use the -target option to specify that SmartUpgrade analyze your application for any features, configuration settings, or artifacts that are specific to 10*g* Release 2 (10.1.2).

If you are upgrading an application that was previously deployed on OC4J as part of an Oracle Application Server 10*g* Release 2 (10.1.2) installation, use the -target option as follows:

```
java -jar smartupgrade.jar --LOCATOR_NAME -target 10.1.2
```

For example:

```
java -jar smartupgrade.jar --ears C:\myApps\my1012App.ear -target 10.1.2
```

3.1.8 Controlling the Output of SmartUpgrade Findings Information and Error Reporting

SmartUpgrade uses both the `stdout` and `stderr` output streams.

SmartUpgrade prints all finding information to `stdout`. You can redirect this output, according to the rules of your operating system and command-line shell, to a file.

Alternatively, users can use the "`-output` *`output_file`*" option to output the stream to a specified file. Consider the following examples:

```
java -jar smartupgrade.jar -output report.txt
java -jar smartupgrade.jar -html -output report.html
```

SmartUpgrade uses the `stderr` stream to report status and diagnostics on the command line. The `-quiet` flag stops SmartUpgrade from outputting anything to `stderr`. When you use the -quiet option, SmartUpgrade does not output any status or diagnostic messages.

The `-quiet` and `-output` options can be used in combination, because the options govern separate, internal behavior characteristics of SmartUpgrade and control how SmartUpgrade produces output.

For related information, see Section 3.1.7.4, "Using the SmartUpgrade Command-Line Interface to Generate an HTML Report".

3.2 Integrating SmartUpgrade with Apache Ant

If you use Apache Ant in your development environment, you can use the custom Ant task shown in Example 3–2 to integrate SmartUpgrade with your existing Ant environment:

Example 3–2 Custom Ant Task for SmartUpgrade

```
<taskdef
   name="SmartUpgrade"
   classname="oracle.smartupgrade.UpgradeTask"
   classpath="${basedir}/smartupgrade.jar"/>
```

After you define the `taskdef`, as shown in Example 3–2, then you can execute SmartUpgrade from within an Ant script.

Example 3–3 shows a typical example, which recursively locates all EAR files in the `demo` directory and executes SmartUpgrade to examine each file. The valid values used for the `upgrade locator` element in Example 3–3 are identical to those used for the LOCATOR_NAME on the Java command line.

For more information, see Section 3.1.5.2, "Identifying a SmartUpgrade Locator".

Example 3–3 Using the SmartUpgrade Custom Ant Task Within Apache Ant

```
<target name="test" depends="declare">
   <SmartUpgrade flags="-quiet"
      <upgrade locator="ears">
         <fileset dir="${basedir}/demo">
            <include name="**/*.ear" />
         </fileset>
      </upgrade>
   </SmartUpgrade>
</target>
```

4

Using SmartUpgrade Generated Artifacts

In addition to generating an upgrade report, SmartUpgrade can generate artifacts, such as the following:

- Web application deployment descriptor files
- Data source configuration files
- The mapping files and type classes required to run Web Services on Oracle WebLogic Server
- EJB configuration files required to deploy your EJB artifacts on Oracle WebLogic Server

For the generated deployment descriptors, you can review the generated Oracle WebLogic Server elements and use them as a starting point for upgrading your application so you can deploy it successfully on Oracle WebLogic Server.

The following sections describe how to use artifacts generated by SmartUpgrade to upgrade your applications so they can be deployed on Oracle WebLogic Server:

4.1 Overview of the Steps Required to Use the SmartUpgrade Generated Artifacts

In most cases, the process of upgrading an application from Oracle Containers for Java EE (OC4J) to Oracle WebLogic Server 11*g* is an iterative process:

1. You run SmartUpgrade once to generate a report and some initial Oracle WebLogic Server artifacts.
2. You review the report, make changes, and then run SmartUpgrade again.
3. You repeat the process until the application is ready to deploy on Oracle WebLogic Server.

Figure 4–1 provides a flow chart of the steps you can use to analyze an application, review the report, and make iterative changes before finally deploying the application on Oracle WebLogic Server.

Table 4–1 provides a description of each step shown in the flow chart.

Figure 4–1 Flow Chart of the SmartUpgrade Process

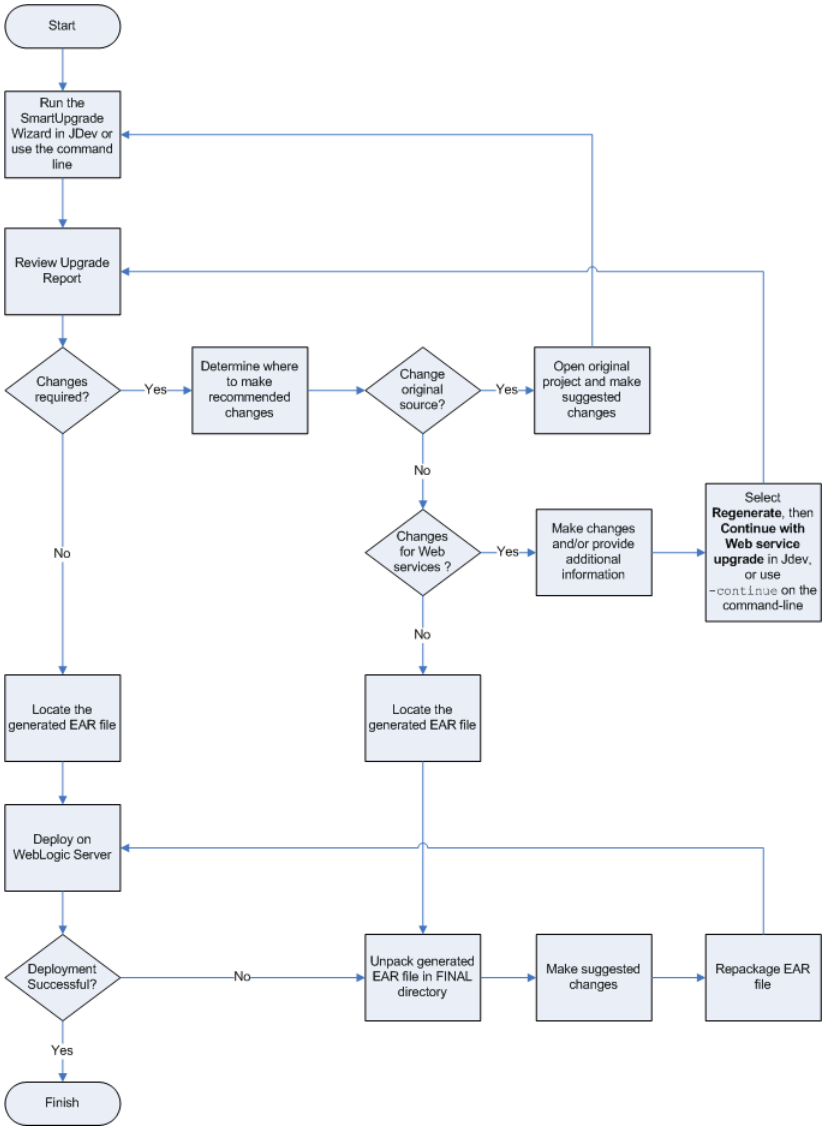


Table 4–1 Description of the Steps in the SmartUpgrade Process Flow Chart

Step	Description
Run the SmartUpgrade Wizard in JDev or use the command line	From Oracle JDeveloper, start the Java EE Upgrade wizard, as described in Section 2.3, "Starting and Using the Java EE Upgrade Wizard" From the command-line interface, run the command-line arguments, as described in Chapter 3, "Using the SmartUpgrade Command Line".

Table 4–1 (Cont.) Description of the Steps in the SmartUpgrade Process Flow Chart

Step	Description
Review Upgrade Report	If you are using Oracle JDeveloper, refer to Section 2.4, "Using a SmartUpgrade Report". If you are using the command-line, open the resulting report in a text editor, or use the appropriate command-line arguments to generate an HTML report that you can review in a Web browser. For more information, see Section 3.1.7.4, "Using the SmartUpgrade Command-Line Interface to Generate an HTML Report". In either case, the finding should indicate whether any artifacts were generated for the finding. This information can be useful later when you are deciding how to implement suggested changes to your application.
Determine where to make the changes	In some cases, the report findings include suggestions on modifications you must make to your application before it will deploy successfully on Oracle WebLogic Server. Some of the identified changes are better made in the original applications; others can be made easier by starting with the files generated by SmartUpgrade. Note that each finding in the report indicates whether or not any artifacts were generated for the finding.
Open original project and make suggested changes	If the findings recommend making changes that you can perform on the original sources, open the original project and make the modifications there.
Make changes and/or provide additional information for Web services	If the findings or the SmartUpgrade output messages identify any Web services-specific errors or recommendations, you might be asked to make the changes and then regenerate or continue the upgrade process. For example, if you are using Oracle JDeveloper, you might see a finding called "Web Service Artifact Generation 10.1.3" in the Upgrade report. The finding recommends that you select **Regenerate** from the Upgrade Report context menu and select specific artifact generation options. You will also see a warning message in the SmartUpgrade log window.
Select **Regenerate**, then **Continue with Web service upgrade** in Jdev, or use `-continue` on the command-line	For more information, see the following: ■ If you are using Oracle JDeveloper, see Section 2.3.4, "Continuing with the Upgrade When Generating Artifacts". ■ If you are using the SmartUpgrade command-line interface, see the information about the `-continue` argument in Table 4–1.

Locate the generated EAR file	When SmartUpgrade generates artifacts, it organizes them into a valid EAR file structure and archives a new EAR file. The EAR file is located in the following directory in your new project directory:
	■ If you are using Oracle JDeveloper and you are using the default output directory for SmartUpgrade:
	`gen_src/`*`app_name`*`_ear.d/final/`
	■ If you are using the command-line interface:
	`upgrade/`*`app_name`*`_ear.d/final/`
	For more information, see Appendix A, "Output Directories Generated by SmartUpgrade".

Table 4–1 (Cont.) Description of the Steps in the SmartUpgrade Process Flow Chart

Step	Description
Unpack generated EAR file in FINAL directory	If you unpack the generated EAR file, you can review the generated artifacts to see the changes that were made to your files in order to deploy them on Oracle WebLogic Server. You can attempt to deploy this EAR file on Oracle WebLogic Server "as is" or you can use the contents as a starting point for additional modifications, by reviewing or modifying the individual artifacts in the EAR file. Note that if you are using Oracle JDeveloper, you can also browse the contents of the EAR file via the Application Navigator.
Make suggested changes	If you have yet to deploy your application on Oracle WebLogic Server, then you can use this step to make changes suggested by the findings in the SmartUpgrade report. If you received errors while attempting to deploy the application EAR file on Oracle WebLogic Server, you can use this step to make any changes suggested by the Oracle WebLogic Server deployment errors.
Repackage EAR file	After you make changes to the unpacked EAR file that was generated by SmartUpgrade, you can then re-archive it manually using the appropriate Java command or other utility.

4.2 Additional Information About the Web Services Artifacts Generated by SmartUpgrade

The following sections provide additional information about the Web services artifacts generated by SmartUpgrade:

- Differences Between OC4J and Oracle WebLogic Server Web Services
- Capabilities of the SmartUpgrade Web Services Artifact Generation
- Analyzing the Performance Impact of the Web Services Glue Code Generated by SmartUpgrade

4.2.1 Differences Between OC4J and Oracle WebLogic Server Web Services

For general information about the differences between OC4J and Oracle WebLogic Server Web services, refer to the following resources:

- "Task 6: Upgrade the Application Web Services" in the *Oracle Fusion Middleware Upgrade Guide for Java EE*
- "Overview of WebLogic Web Services" in *Oracle Fusion Middleware Introducing WebLogic Web Services for Oracle WebLogic Server*

4.2.2 Capabilities of the SmartUpgrade Web Services Artifact Generation

When you generate Oracle WebLogic Server Web services artifacts, SmartUpgrade upgrades the following OC4J Web services to Oracle WebLogic Server 10.3 JAX-RPC Web services:

- OC4J 10*g* Release 3 (10.1.3) Web services
- OC4J 10*g* Release 3 (10.1.3) EJB 2.1 Web services
- OC4J 10*g* Release 3 (10.1.3) EJB 3.0 Web services

- OC4J 10*g* Release 2 (10.1.2) Web services

The upgraded Web services use the same `contextpath` and the context URI as the OC4J 10*g* Web services. For example, suppose the URI for your OC4J 10*g* Release 3 (10.1.3) Web services were as follows:

```
http://locahost:8888/contextpath/soap11
http://locahost:8888/contextpath/soap12
```

After the upgrade, you should be able to access the same Web services via the following URI on the Oracle WebLogic Server domain:

```
http://locahost:7001/contextpath/soap11
http://locahost:7001/contextpath/soap12
```

During the upgrade process, SmartUpgrade automatically updates Web services related deployment descriptors (such as `web.xml` and `weblogic-webservices.xml`) and generates "glue code" that allows your upgraded Web services to communicate with your existing remote endpoints.

For more information about the Web services upgrade tasks that SmartUpgrade performs for you, see Section 1.4.2, "Generation of Web Services Artifacts".

You can optionally configure SmartUpgrade to analyze your application specifically to determine the performance impact of the generated glue code. For more information, see Section 4.2.3, "Analyzing the Performance Impact of the Web Services Glue Code Generated by SmartUpgrade".

4.2.3 Analyzing the Performance Impact of the Web Services Glue Code Generated by SmartUpgrade

To upgrade your OC4J Web services to Oracle WebLogic Server, SmartUpgrade generates Java programming code called "glue code," which allows your upgraded Web services to continue to use your existing business logic code while deployed on the new server.

To determine the performance impact of the generated glue code:

1. Select one of the following options when you use SmartUpgrade to upgrade your application and generate Web services artifacts:
   - If you are using Oracle JDeveloper, select **Generate Instrumented Code for Performance Analysis** on the OC4J Web Services Upgrade page of the Java EE Upgrade wizard.
   - If you are using the SmartUpgrade command-line interface, use the `-evaluate` command-line option.

   If you enable this option, then SmartUpgrade instruments the generated wrapper ("glue") code for performance analysis.

   Note that this option is appropriate only for testing the application, and not for production environments.
2. Perform the upgrade of your application and deploy the application on Oracle WebLogic Server.
3. After the application is deployed and running successfully on Oracle WebLogic Server, locate the following HTML file in the Oracle WebLogic Server domain directory:

   MW_HOME`/user_projects/domains/`*domain_name*`/`*webservice_port_name*`.html`

4. Open the HTML file in a browser.

   The browser displays the Glue Code Performance Analysis page. Immediately after deployment, no data is displayed in each column of the table.

5. Exercise the Web service to generate some performance data.

   You can test the Web service using the Oracle WebLogic Server Administration Console or the Oracle Enterprise Manager Fusion Middleware Control.

   For more information, see "Testing Your Web Services" in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

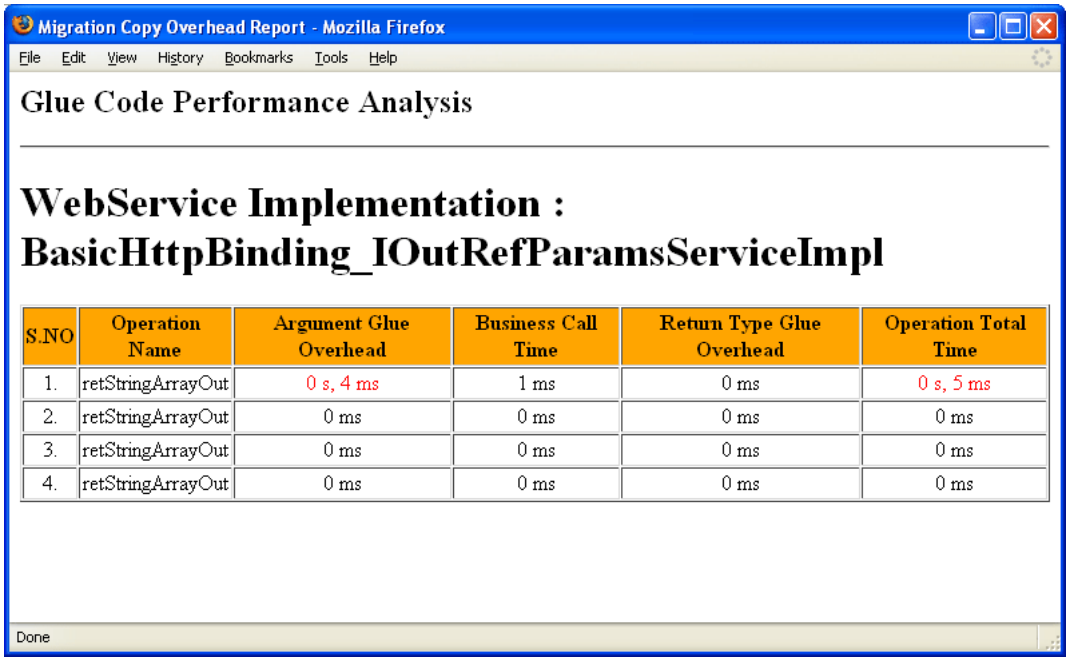
6. Refresh the browser where the Glue Code Performance Analysis page is displayed.

   Each time you test the Web service, an entry is added to the Glue Code Performance page. Use the data on this page to measure the impact of the glue code on the performance of the Web service.

Note: When you are finished analyzing the performance metrics, regenerate the application for production environment, by running SmartUpgrade again with the following options:

- If you are using Oracle JDeveloper, then repeat the process from the start , but do not select **Generate Instrumented Code for Performance Analysis**.
- If you are using the command-line interface, then use the `-evaluate false` option when you run SmartUpgrade.

Figure 4–2 SmartUpgrade Web Services Upgrade - Glue Code Performance Analysis Page



A

Output Directories Generated by SmartUpgrade

This appendix describes the output directories generated when you upgrade an application using SmartUpgrade.

By default, SmartUpgrade saves the generated artifacts to a series of directories on disk. If you are using Oracle JDeveloper, then the directories are relative to the project directory. If you are using the command-line interface, they are relative to the directory where you are running the command-line interface.

More specifically:

- If you are using Oracle JDeveloper, then the output directories are by default under the `/gen_src/` directory inside your new project directory. However, you can define a different location from the OC4J Artifacts page of the Java EE Upgrade wizard. The location is relative to the project directory where SmartUpgrade saves the upgrade report.
- If you are using the SmartUpgrade command-line interface, the output directories are created in the upgrade directory inside the directory where you invoked the command-line tool.

Figure A–1 illustrates the organization of the output files on disk.

Table A–1 provides a description and purpose of the key directories that are generated.

Table A–2 describes the output directories specific to Web services artifact generation.

Figure A–1 Output Directories for Generated Artifacts

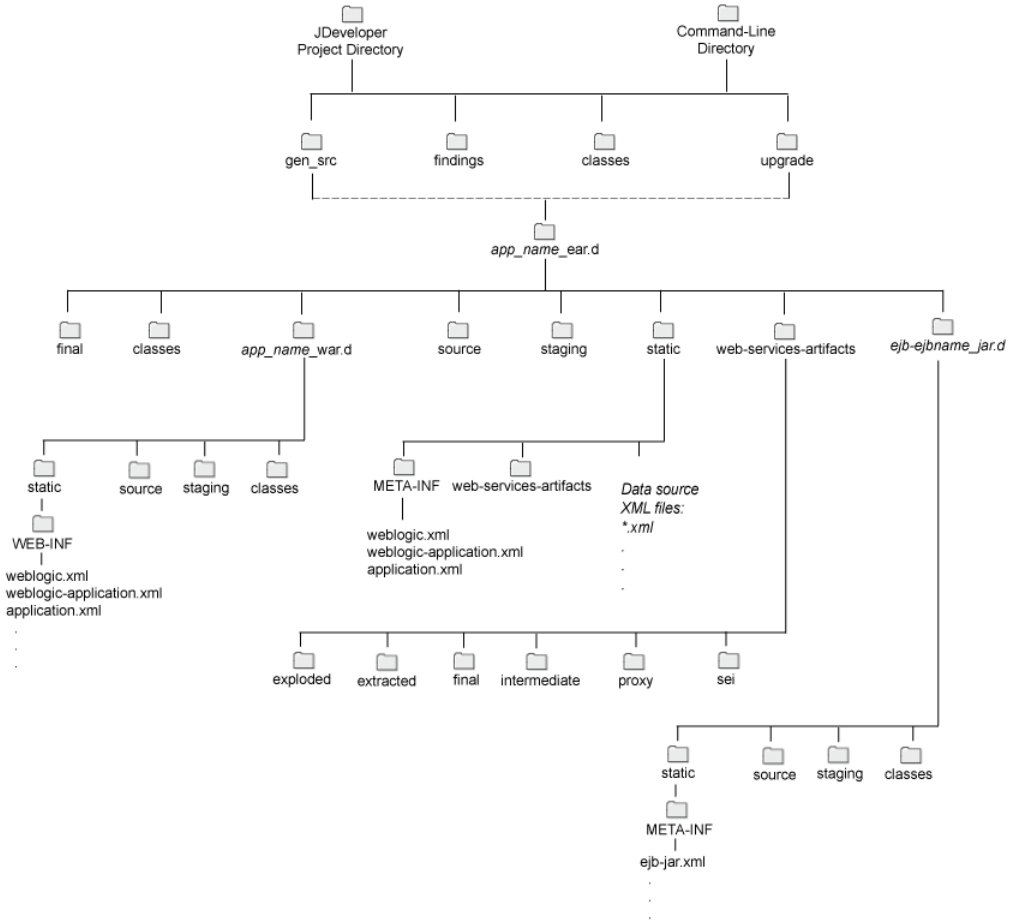


Table A–1 Summary of the SmartUpgrade Output Directories

Directory	Description
final	This directory contains the archive that represents the final output of the SmartUpgrade tool. In the case of the root directory (in the *app_name*_ear.d directory), it represents the complete archive that contains all the SmartUpgrade-generated artifacts.
classes	This directory contains the class files used to compile and generate the artifacts.
source	This directory contains the source files used to compile and generate the artifacts.
staging	This is a directory used to stage the subcomponents of the application before archiving the entire application into the final directory within the root directory.
static	This directory contains static artifacts , such as data-source configuration files that are generated by SmartUpgrade.

Table A–2 Summary of the Output Directories Specific to Web Services

Directory	Description	When is it created or overwritten?
final	This folder contains the final, upgraded, and deployable Web services application. The application is expanded on disk, as well as in the form of a packaged enterprise archive file.	This directory gets overwritten and recreated each time you run SmartUpgrade with the options for generating Web services artifacts. Specifically, the directory is overwritten if you regenerate the artifacts using the Regenerate option on the SmartUpgrade Report context menu or if you use the continue command-line argument. The directory is recreated by merging the content of the **exploded** and **intermediate** directories.
intermediate	This directory contains the upgraded Web services artifacts and glue code that are not yet packaged in the final deployable application archive. It contains intermediate information only for Web service artifacts.	This directory gets overwritten and recreated each time you run SmartUpgrade when it is configured to generate Web services glue code. You can prevent SmartUpgrade from overwriting this directory by enabling the option to skip the generation of glue code. Then you can make additional changes by hand to the generated code.
exploded	This directory contains the original, input application archive, as-is, exploded on disk, with no changes.	This directory gets overwritten and is recreated each time you run SmartUpgrade when it is configured to generate a source upgrade plan.

Index

A

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z