Sun™ Identity Manager 8.0 Upgrade

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820-2963-10

Contents

List of Tables . . . vii

Preface . . . ix

Who Should Use This Book . . . ix
Conventions Used in This Book . . . x
  Typographic Conventions . . . x
  Symbols . . . xi
  Shell Prompts . . . xi
Accessing Sun Resources Online . . . xiii
Contacting Sun Technical Support . . . xiii
Related Third-Party Web Site References . . . xiii
Sun Welcomes Your Comments . . . xiv

Chapter 1 Overview of the Upgrade Process . . . 1
Why Upgrade? . . . 1
Upgrade Paths and Support Policies . . . 1
  Identity Manager Upgrade Paths . . . 2
  End of Service Life for Software Support . . . 3
    Full Support Phase . . . 4
    Limited Support Phase . . . 4
  Identity Manager's Deprecation Policy . . . 4
How Easy is Upgrading? . . . 5
Terminology and Concepts . . . 7
  Identity Manager Product Versus Identity Manager Application . . . 7
  Development, Test, QA and Production Environments . . . 7
  Promotion . . . 8
  Source-Control and CBE . . . 8
  Baseline and Image . . . 9
  Skip-Level Upgrade . . . 9
  Configurations and Customizations . . . 10

Upgrade Process and Upgrade Procedure 10
Upgrade Project 11
Phases of the Upgrade Project 11

Chapter 2 Planning 13
Task 1: Review Your Production Environment 13
Step 1: Document Your Platform 14
Application Servers 15
Database Servers 15
Sun Identity Manager Gateway 15
Java Runtime Environment 15
Supported Resources 15
Web Servers 15
Step 2: Document Your Identity Manager Installation 16
Identity Manager Version 16
Identity Manager Assessment Tools 16
Step 3: Document Your Custom Components 18
Customized Database Table Definitions 18
Custom Filesystem Objects 18
Custom Repository Objects 20
Task 2: Choose the Target Identity Manager Version 22
Task 3: Prepare Your Test Plan 24
Review Your Existing Test Plan 24
Create a Test Plan 24
Task 4: Prepare Your Upgrade Procedure 25
Review Your Existing Upgrade Procedure 25
Create an Upgrade Procedure 26

Chapter 3 Developing and Testing Your Upgrade 27
Task 5: Reset Your Development Environment 27
Task 6: Upgrade Your Development Environment 28
Step 1: Stop Active Sync and Reconciliation 29
Step 2: Stop the Identity Manager Application 29
Step 3: Back Up Your Identity Manager Application 29
Step 4: Remove Hotfixes 29
Step 5: Take a Snapshot 30
Step 6: Update Your Platform 31
Step 7: Upgrade the Identity Manager Product 32
Update the Repository Database Tables 32
Upgrade the Identity Manager Product 32
Troubleshooting Upgrade 33
To Install on a Windows Platform 36

To Install on a Unix Platform 38
Upgrade Gateway and Password Sync 39
Step 8: Take Another Snapshot 39
Step 9: Analyze the Changes 40
Restoring Customized Files 41
Restoring Customized Objects 42
Step 10: Rebuild Any Custom Java 42
Step 11: Make Necessary Changes in XPRESS 43
Step 12: Test Your Identity Manager Application in the Development Environment 43
Step 13: Restart Active Sync and Reconciliation 44
Step 14: Merge Changes Back into Source-Control 44
Task 7: Reset Your Test Environment 45
Step 1: Reset Your Platform to Match Production 45
Step 2: Set Up for Functional Testing 45
Task 8: Execute Your Upgrade Procedure 46
Step 1: Stop Active Sync and Reconciliation 46
Step 2: Stop the Identity Manager Application 46
Step 3: Back Up Your Identity Manager Application 47
Step 4: Remove Hotfixes 47
Step 5: Change `TaskDefinition` Objects 47
Step 6: Update Your Platform 48
Step 7: Upgrade Your Identity Manager Application 49
Update Your Database Table Definitions 50
Promote the Identity Manager Application 50
Import a Subset of `update.xml` 50
Upgrade Your Gateway and Password Sync Components 50
Step 8: Test Your Identity Manager Application 51
Step 9: Restart Active Sync and Reconciliation 51
Step 10: Restart the Identity Manager Application 52
Task 9: Perform Functional Testing 52

Chapter 4 User Acceptance Testing 53
Task 10: Reset Your QA Environment 54
Task 11: Upgrade Your QA Environment 55
Task 12: Perform User Acceptance Testing 55

Chapter 5 Upgrading Your Production Environment 57
Special Considerations for Production 57
Task 13: Upgrade Your Production Environment 58

Appendix A Skip-Level Upgrade Considerations 59
Overview 59
Phases of a Skip-Level Upgrade 61
Task 2: Choose the Target Identity Manager Version (Enhanced) 63
Task 5: Upgrade Your Development Environment (Repeated) 64
    Step 9: Analyze the Changes for a Skip-Level Upgrade 64

Appendix B Assessment Worksheets 67
Platform Inventory 68
Identity Manager Installation 71
Custom Components 72

Index 75

List of Tables

Table 1 Typographic Conventions x
Table 2 Symbol Conventions xi
Table 3 Shell Prompts xi
Table 1-1 End of Service Life (EOSL) for Software Support 3
Table 2-1 `installed` Utility Options 17
Table 2-2 `inventory` Utility Options 17
Table 3-1 `savedFiles` Directory File Structure 41
Table B-1 Inventory Information 68
Table B-2 Resource Information 69
Table B-3 Identity Manager Installation Information 71
Table B-4 Custom Work Information 72

Preface

This guide provides detailed information and instructions to help you upgrade the Sun™ Identity Manager (Identity Manager) product.

NOTE If your current Identity Manager installation has a large amount of custom work, contact your Sun Professional Services about getting assistance with your upgrade.

Who Should Use This Book

Sun™ Identity Manager Upgrade was designed for deployers and system administrators who are responsible for upgrading and configuring Identity Manager and associated software in a Production environment.

How This Book Is Organized

Identity Manager Upgrade is organized into these chapters:

- Chapter 1, "Overview of the Upgrade Process," provides an overview of the upgrade process.
- Chapter 2, "Planning," helps you identify goals and gather information to prepare for upgrading Identity Manager.
- Chapter 3, "Developing and Testing Your Upgrade," provides guidelines for developing and verifying your upgrade procedure.
- Chapter 4, "User Acceptance Testing," describes how to test your upgrade in an environment that simulates your Production environment.

- Chapter 5, "Upgrading Your Production Environment," provides information and suggestions for upgrading your Production environment.
- Appendix A, "Skip-Level Upgrade Considerations," describes extra steps you must perform to upgrade Identity Manager more than one version at a time.
- Appendix B, "Assessment Worksheets," provides worksheets that help you gather information in preparation for upgrading.

Conventions Used in This Book

The tables in this section describe the conventions used in this book including:

- Typographic Conventions
- Symbols
- Shell Prompts

Typographic Conventions

The following table describes the typographic conventions used in this book.

Table 1 Typographic Conventions

Typeface	**Meaning**	**Examples**
`AaBbCc123` (Monospace)	API and language elements, HTML tags, Web site URLs, command names, file names, directory path names, on-screen computer output, sample code.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123` (Monospace bold)	What you type, when contrasted with onscreen computer output.	`%` **`su`**
		`Password:`

AaBbCc123 (Italic)	Book titles, new terms, words to be emphasized. A placeholder in a command or path name to be replaced with a real name or value.	Read Chapter 6 in the *User's Guide*. These are called *class* options. Do *not* save the file. The file is located in the *install-dir*`/bin` directory.

Symbols

The following table describes the symbol conventions used in this book.

Table 2 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional command options.	`ls [-l]`	The `-l` option is not required.
`-`	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
`+`	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
`>`	Indicates menu item selection in a graphical user interface.	File > New > Templates	From the File menu, choose New. From the New submenu, choose Templates.

Shell Prompts

The following table describes the shell prompts used in this book.

Table 3 Shell Prompts

Shell	Prompt
C shell on UNIX or Linux	*machine-name*`%`
C shell superuser on UNIX or Linux	*machine-name*`#`
Bourne shell and Korn shell on UNIX or Linux	`$`
Bourne shell and Korn shell superuser on UNIX or Linux	`#`
Windows command line	`C:\`

Related Documentation and Help

Sun provides additional printed and online documentation and information to help you install, use, and configure Identity Manager:

- *Identity Manager Installation*: Step-by-step instructions and reference information to help you install and configure Identity Manager and associated software.
- *Identity Manager Administration*: Procedures, tutorials, and examples that describe how to use Identity Manager to provide secure user access to your enterprise information systems.
- *Identity Manager Technical Deployment Overview*: Conceptual overview of the Identity Manager product (including object architectures) with an introduction to basic product components.
- *Identity Manager Deployment Tools*: Reference and procedural information that describes how to use different Identity Manager deployment tool. This information addresses rules and rules libraries, common tasks and processes, dictionary support, and the SOAP-based Web service interface provided by the Identity Manager server.
- *Identity Manager Workflows, Forms, and Views*: Reference and procedural information that describes how to use the Identity Manager workflows, forms, and views — including information about the tools you need to customize these objects.
- *Identity Manager Resources Reference*: Reference and procedural information that describes how to load and synchronize account information from a resource into Sun Java™ System Identity Manager.
- *Identity Manager Tuning, Troubleshooting, and Error Messages*: Reference and procedural information that provides guidance for tuning Sun Java™ System Identity Manager, provide instructions for tracing and troubleshooting problems, and describe the error messages and exceptions you might encounter as you work with the product.
- *Identity Manager Service Provider Deployment*: Reference and procedural information that describes how to plan and implement Sun Java™ System Identity Manager Service Provider.
- *Identity Manager Help*: Online guidance and information that offer complete procedural, reference, and terminology information about Identity Manager. You can access help by clicking the Help link from the Identity Manager menu bar. Guidance (field-specific information) is available on key fields.

Accessing Sun Resources Online

For product downloads, professional services, patches and support, and additional developer information, go to the following:

- Download Center
  `http://wwws.sun.com/software/download/`
- Professional Services
  `http://www.sun.com/service/sunps/sunone/index.html`
- Sun Enterprise Services, Solaris Patches, and Support
  `http://sunsolve.sun.com/`
- Developer Information
  `http://developers.sun.com/prodtech/index.html`

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, contact customer support using one of the following mechanisms:

- The online support Web site at `http://www.sun.com/service/online/us`
- The telephone dispatch number associated with your maintenance contract

Related Third-Party Web Site References

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document.

For example, the title of this book is *Sun Java™ System Identity Manager 8.0 Upgrade*, and the part number is *820-2963-10*.

Chapter 1

Overview of the Upgrade Process

This chapter provides an overview of the Identity Manager upgrade process, which includes the following topics:

- Why Upgrade?
- Upgrade Paths and Support Policies
- How Easy is Upgrading?
- Terminology and Concepts
- Phases of the Upgrade Project

Why Upgrade?

Upgrading your release of Identity Manager provides several advantages, including:

- Access to advanced features and functionality
- A more secure environment for your servers
- Continued eligibility for full support and services

Upgrade Paths and Support Policies

This section provides the following information:

- Identity Manager Upgrade Paths
- End of Service Life for Software Support
- Identity Manager's Deprecation Policy

Identity Manager Upgrade Paths

To determine the upgrade path you must follow when upgrading to a newer version of Identity Manager, see “Upgrade Paths and Support Policies” in the *Sun Java™ System Identity Manager 8.0 Release Notes*. Generally, use the latest available patch or service pack for the version to which you are upgrading.

Using non-standard upgrade techniques is *strongly* discouraged because serious, non-obvious consequences can result.

Identity Manager's standard upgrade process performs steps that are necessary to convert existing repository objects into the format that the newer version of Identity Manager expects. In particular, the *updater* programs that are shipped with each version of Identity Manager contain special logic that preserves the original meaning and behavior of these objects. Each updater program updates a specific type of configuration object, and each updater is invoked by an `ImportCommand` within `update.xml` or within a file that `update.xml` includes. Updated versions of pre-existing objects often use slightly different mechanisms than those used in the sample objects that ship with the new Identity Manager version.

NOTE If you are upgrading Identity Manager, you must follow the required upgrade path noted in the “Upgrade Paths and Support Policies” section of the *Identity Manager Release Notes*.

Even if you plan to perform a clean installation of Identity Manager and migrate your customizations to the new Identity Manager version, you must apply the standard upgrade process for each version of Identity Manager to properly update existing repository objects. The updater programs that are shipped with each version of Identity Manager work *only* with that version.

Simply comparing the objects that are produced by importing the new `init.xml` with the objects that are produced by importing the old `init.xml` file is not a sufficient way to detect the changes that must be made in order to upgrade existing repository objects. Comparing these two sets of objects is a quick way to estimate the effort associated with an upgrade, but the standard upgrade path is the best way to achieve a safe and complete upgrade.

End of Service Life for Software Support

During the End of Service Life (EOSL) period, Identity Manager software support is offered in two phases:

- *Phase 1: Full Support*
- *Phase 2: Limited Support*

NOTE The length of the Full Support Phase varies by product.

The following table provides information about the EOSL and EOL dates for older versions of Identity Manager.

Table 1-1 End of Service Life (EOSL) for Software Support

Product Name	Product Status	Last Ship Date	Phase 1 End Date	Phase 2 End Date (EOSL)	EOL Announcement
Sun Java System Identity Manager 7.1	Post-RR				
Sun Java System Identity Manager 7.0	Post-RR				
Sun Java System Identity Manager 6.0 2005Q4	Post-RR	05/25/2007	05/25/2008	05/2012	11/20/06
Sun Java System Identity Auditor 1.0 2005Q1	Post-RR	02/02/2007	02/2008	02/2012	08/01/06
Sun Java System Identity Manager Service Provider Edition 1.0 2005Q3	Post-RR	02/02/2007	02/2008	02/2012	08/01/06
Sun Java System Identity Manager 5.0 2004Q3	EOL	08/11/2006	08/2007	08/2011	02/07/06
Sun Java System Identity Manager 5.0 SP*x* 2004Q3	EOL	08/11/2006	08/2007	08/2011	02/07/06
Sun Java System Identity Manager 5.5	EOL	08/11/2006	08/2007	08/2011	02/07/06
Waveset Lighthouse 4.1			03/2006	03/2010	

Full Support Phase

During the Full Support Phase, Sun Microsystems, Inc. provides software support in accordance with the customer's support contract with Sun (including the applicable Service Listing) as set forth at:

http://www.sun.com/service/servicelist/

As of the software product's announced EOL date, customers will no longer have access to software updates and upgrades for that product.

Limited Support Phase

During the Limited Support Phase, Sun Microsystems, Inc. provides software support in accordance with the customer's support contract with Sun (including the applicable Service Listing) as set forth at:

http://www.sun.com/service/servicelist/

However, customers are not entitled to submit bugs or to receive new patches from Sun Microsystems, Inc. As with Full Support Phase, after the software product's announced EOL date, customers will no longer have access to software updates and upgrades for that software product.

Identity Manager's Deprecation Policy

Generally, Identity Manager Engineering does not remove interfaces, public classes, or public methods without first announcing the change and then waiting at least one full year before removing the interface, class, or method.

NOTE Deprecations are announced in the *Identity Manager Release Notes.*

When upgrading, always check the Release Notes for the new Identity Manager version to get the latest information about deprecated features.

For example, if Identity Manager Engineering decides to remove a public Java class from the product or change the signature or behavior of public methods in a public class, Engineering deprecates that class or method, and then waits at least one year. Additionally, when method signatures or behaviors must be changed, the standard policy is to add a new method and deprecate the old method.

Java classes and methods are deprecated programmatically using the `@deprecated` Javadoc tag. This `@deprecated` tag causes the Java compiler to emit a warning message when customers recompile code that refers to a deprecated class or method. Both the `@deprecated` tag and the deprecation warning message provide

- The Identity Manager release in which the class or method was deprecated
- Any class or method that replaces the deprecated class or method

The `@deprecated` tag does not work for other types of interfaces and behavior. For instance, there is no way to use the `@deprecated` tag for JSPs, Views, or configuration objects. You must check the Release Notes for this information because Engineering cannot deprecate these interfaces programmatically.

Identity Manager Engineering generally removes deprecated interfaces or behavior no sooner than one full year after the release in which the class or method was first deprecated. Engineering generally removes deprecated interfaces or behavior only in a full release of Identity Manager. A full release is a major or a minor release (for example, 8.0 or 8.1).

Exceptions to this policy might be necessary in unusual circumstances, but Identity Manager Engineering is committed to providing backward-compatibility for its customers to the extent that it is feasible to do so.

How Easy is Upgrading?

Basically, how easy or how difficult it is to upgrade Identity Manager depends on how much you customize Identity Manager and what kind of customizations you make. Some customizations are highly backward-compatible. For example, older workflows, forms, and rules tend to work without modification on newer versions of Identity Manager. Other customizations are more volatile, and require additional work to integrate them with new versions of Identity Manager. For example, you might have to recompile or rewrite integration code that invokes Java classes that are part of the Identity Manager product.

The most significant effect of customization, however, is that customization changes how you must approach upgrading Identity Manager. Note that the following discussion uses terminology and concepts discussed on page 7.

Upgrading Identity Manager can be relatively simple if you use the product as shipped — that is, if you *configure* Identity Manager but do not *customize* it. If you are unfamiliar with these terms, see "Configurations and Customizations" on page 10. The upgrade process that is part of each version of the Identity Manager product was originally designed to be run in each environment. The product upgrade process automatically preserves your configuration settings as it updates Identity Manager's configuration objects and other repository objects to work with the newer version of code. However, the upgrade process does not know how to update your customizations. In fact, the upgrade process simply ignores most customizations. The upgrade process does save off the customized version of any file that the upgrade process intends to replace, but this is the most that the upgrade process can do automatically.

If you customize Identity Manager, as the vast majority of customers do, then you cannot simply run the standard Identity Manager product upgrade in each environment. You must maintain and must re-test your customizations after you upgrade the Identity Manager product, and you want to be certain that exactly what you tested goes into each environment. Because Identity Manager is often customized extensively, those who deploy Identity Manager have developed sophisticated practices for managing their customizations. For more information, see "Source-Control and CBE" on page 8.

If you customize Identity Manager, the most common approach is to manage your customizations by maintaining a baseline and then to generate and promote an image of your Identity Manager application into each target environment. See "Baseline and Image" on page 9. You apply the standard Identity Manager product upgrade process only in your Development environment. You then apply to your application baseline most of the changes that the upgrade makes. See "Identity Manager Product Versus Identity Manager Application" on page 7. However, your upgrade procedure also must include some pieces of the Identity Manager product upgrade. For example, your upgrade procedure will include some subset of `update.xml` to update repository objects that are not managed as part of your baseline. Your upgrade procedure might also include database table scripts to update Identity Manager's repository table definitions.

Because the vast majority of customers do customize Identity Manager, and because many customize Identity Manager extensively, this document assumes that you have customizations and that you manage your customizations with this approach.

Terminology and Concepts

This section describes some specialized terminology that relates to upgrading Identity Manager.

Identity Manager Product Versus Identity Manager Application

In some sections, this document distinguishes between the Identity Manager *product* and your Identity Manager *application*.

- Identity Manager product means the product as shipped, with standard samples, JSPs, JARs, and configuration objects.
- Identity Manager application means the customer's deployment of Identity Manager, which will be uniquely configured and might be customized, might include custom code and modified JARs, might be re-labeled, and so forth.

Development, Test, QA and Production Environments

This document assumes that you are using the following, different types of environments:

- A Development environment is where you configure, customize, and use source-control to build an image of the Identity Manager application to be promoted to another environment. You also write an upgrade procedure in this environment that you follow in each target environment.
- A Test environment is where you test your upgrade procedure against controlled data and perform controlled testing of the resulting Identity Manager application.
- A QA environment is where you test your upgrade procedure against data, hardware, and software that closely simulate the Production environment and where you allow intended users to test the resulting Identity Manager application.
- A Production environment is where the Identity Manager application is actually available for business use.

Promotion

In this document, *promotion* refers to the process of generating a particular version of your Identity Manager application and moving that version from Development to Test, from Test to QA, and finally from QA to Production.

Source-Control and CBE

You should use source-control tools to manage your configuration settings, any custom configuration objects, any custom code, any test plans, and any automated tests. Any source-control tool, such as CVS or Subversion, should allow you to track changes to any number of individual, text or binary files and to reproduce a particular version of the Identity Manager application. This document refers to these tools generically as *source-control*.

Some of these source-controlled files must be *tailored* for each environment. In particular, configuration objects contain values that often change for each environment. A particular Identity Manager resource object might point to one host machine in your Development environment, to another host machine in your Test environment, and to a third host machine in your Production environment.

Many customers use the Identity Manager IDE plug-ins for NetBeans and Eclipse to generate a tailored version of the Identity Manager application for each environment. The Identity Manager IDE plug-ins include a Configuration Build Environment (CBE). Some customers use older versions of the CBE that predate the Identity Manager IDE plug-in. A few customers use other tools and approaches, including proprietary or home-grown mechanisms. This document refers to these tools generically, and to the Identity Manager IDE specifically, as *CBE*.

The Identity Manager IDE plug-in and older CBE versions parameterize configuration objects (and can parameterize code) by replacing environment-specific values with placeholder values. To generate an image of the Identity Manager application that is appropriate to each environment, these tools replace the placeholder values with configured values that are appropriate to each environment. If you manage the parameterized objects in source-control, you can "build out" the same version of the Identity Manager application for each target environment using a process that is predictable and repeatable.

Baseline and Image

In this document *baseline* refers to a tagged level of artifacts, which includes configuration objects, libraries, custom code in source-control that corresponds to a particular version of your Identity Manager application. Your CBE can generate from this baseline an *image* of the Identity Manager application that is appropriate to each target environment. An image is a complete, working copy of the application that has been tailored for a specific environment.

At a minimum, your source-control must contain a baseline for the version of your Identity Manager application that is currently deployed in your Production environment. Your source-control may also contain baselines for previous versions of your Identity Manager application and baselines for versions of your Identity Manager application that you are still developing in preparation to roll out new functionality (such as modified configurations, modified workflows, modified forms, new custom code for integrations).

Any baseline should be complete enough that you can use it to rebuild the corresponding version of your Identity Manager application. Some customers include the Identity Manager product itself within that baseline. Other customers save the Identity Manager product image separately and use the baseline artifacts to overlay that product image with their own configurations and customizations.

Skip-Level Upgrade

A *skip-level*, or *multi-hop*, upgrade updates your Production environment to an Identity Manager product version that is beyond the next major release from its current version. A skip-level a upgrade typically requires a *series of upgrades*, but it is technically possible, and some customers find it feasible, to develop a single upgrade procedure that updates a target environment directly to the target Identity Manager version.

NOTE Appendix A, "Skip-Level Upgrade Considerations" on page 59 describes special considerations for a multi-hop upgrade. If you are uncertain about how to proceed, or if these considerations seem unclear to you, contact Sun Professional Services for assistance with planning and executing your upgrade.

Configurations and Customizations

In this document, the term *configuration* means editing configuration objects in the Identity Manager product. You can edit configuration objects using Identity Manager's Administrator interface or by editing repository objects in accordance with published documentation. For example, configuration includes specifying values in System Configuration or in a Reconciliation Policy. Configuration also includes defining Resource objects that represent the target systems and applications that Identity Manager will manage.

For the purposes of this document, the term *customization* refers to any code, file, or repository object that you write. Customizations include adding your own Java code, adding or modifying Java Server Pages (JSP) files, and writing your own XPRESS code such as workflow, forms, and rules. Creating your own configuration objects or message catalogs also counts as customization.

Sun Professional Services defines these terms somewhat differently. For example, Sun Professional Services might consider customer-specific workflows to be *configurations*.

In this document, however, the key distinction is that the Identity Manager product upgrade automatically preserves and updates customer-specified values within certain well-known objects and within instances of certain well-known types of objects; these are *configurations*. The Identity Manager product upgrade does not attempt to modify customer-written code or objects; these are *customizations*.

Upgrade Process and Upgrade Procedure

In this document, *upgrade process* refers to the upgrade mechanisms that are part of every version of the Identity Manager product. Each new version of Identity Manager contains not only updated code, but also contains an `update.xml` script. This `update.xml` script carefully preserves your configuration settings as it updates existing repository objects to work with the updated code. Any full release of Identity Manager might also contain sample database table scripts. These sample database table scripts carefully migrate existing data and update your database table definitions to work with the updated code.

This document uses *upgrade procedure* to refer to a document that you develop and maintain. An upgrade procedure often takes the form of a checklist. Your upgrade procedure says exactly what you plan to do in each target environment to upgrade your Identity Manager application. See "Task 4: Prepare Your Upgrade Procedure" on page 25.

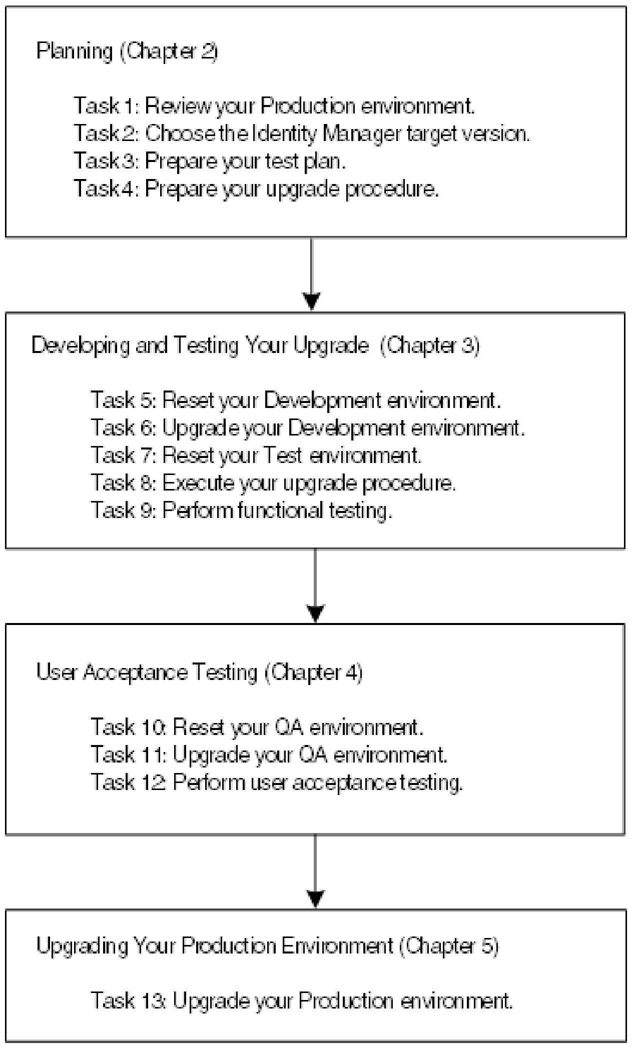
Upgrade Project

This document describes phases in an *upgrade project*. An upgrade project is your effort to upgrade your Identity Manager application to work with a new version of the Identity Manager product on which your application is based. An upgrade project includes your efforts to maintain and re-test your configurations and customizations after executing Identity Manager's upgrade process in your Development environment. An upgrade project also includes your efforts to maintain and re-test the upgrade procedure that you will follow in each target environment.

Phases of the Upgrade Project

The following figure shows the major phases of the upgrade project, summarizing the tasks and steps that you must perform to complete each phase. The rest of this document guides you through each phase.

Figure 1-1 Upgrade Phases


Planning (Chapter 2)
Task 1: Review your Production environment.
Task 2: Choose the Identity Manager target version.
Task 3: Prepare your test plan.
Task 4: Prepare your upgrade procedure.
Developing and Testing Your Upgrade (Chapter 3)
Task 5: Reset your Development environment.
Task 6: Upgrade your Development environment.
Task 7: Reset your Test environment.
Task 8: Execute your upgrade procedure.
Task 9: Perform functional testing.
User Acceptance Testing (Chapter 4)
Task 10: Reset your QA environment.
Task 11: Upgrade your QA environment.
Task 12: Perform user acceptance testing.
Upgrading Your Production Environment (Chapter 5)
Task 13: Upgrade your Production environment.

Chapter 2

Planning

Careful preparation allows for a smoother upgrade. Listing your goals for the upgrade can help you make decisions that are appropriate for your company's needs.

The Planning phase of the upgrade process includes the following tasks:

Task 1: Review Your Production Environment

Task 2: Choose the Target Identity Manager Version

Task 3: Prepare Your Test Plan

Task 4: Prepare Your Upgrade Procedure

Task 1: Review Your Production Environment

Upgrading to a newer Identity Manager release might require changes to the platform in your environment. You can determine the best upgrade path and estimate the complexity of the upgrade by assessing and documenting your Production environment.

This section describes the steps you perform when reviewing your Production environment:

Step 1: Document Your Platform

Step 2: Document Your Identity Manager Installation

Step 3: Document Your Custom Components

➤ **Best Practice**

If you use source-control and CBE to manage this information, these can serve as the documentation for your Identity Manager installation and for your custom components. Review the information and familiarize yourself with the various environments in which you deploy your Identity Manager application, giving particular attention to your Production environment.

Step 1: Document Your Platform

To determine the best upgrade path, use the worksheets provided in Appendix B, "Assessment Worksheets" to inventory the components of your current platform, including:

- Application Servers
- Database Servers
- Sun Identity Manager Gateway
- Java Runtime Environment
- Supported Resources
- Web Servers

NOTE Verify that you are using the correct version of these components for the upgrade version you want to install. Check the "Supported Software and Environments" sections in the *Identity Manager Installation* and *Identity Manager Release Notes* provided for the Identity Manager version to which you want to upgrade.

CAUTION If you are using an Oracle repository, the Identity Manager 8.0 repository DDL uses data types that are not properly handled by older Oracle JDBC drivers. The JDBC drivers in `ojdbc14.jar` do not properly read all of the columns in the log table.

You must upgrade to the `oracle11g_jdbc.jar` drivers for Identity Manager to work properly.

Application Servers

Record the application server version and note any additional patches or service packs. In addition, record the following:

- Operating system version and note any additional patches or service packs
- Java Development Kit (JDK) version required by your application server

Database Servers

Record the database server version and note any additional patches or service packs.

Sun Identity Manager Gateway

Verify which Sun Identity Manager Gateway version you are running by performing the following steps:

1. Open a command window and execute the following command on each of the Gateway servers:

   ```
   gateway -v
   ```

2. Record the results.
3. Record the operating system version of each Gateway server.

NOTE The Gateway server version should always be the same as the Identity Manager version.

Java Runtime Environment

Record the currently installed JRE version required by the `lh` console.

Supported Resources

Record supported resources names, versions, and note any additional patches or service packs.

Web Servers

Record the Web server version and note any additional patches or service packs.

Step 2: Document Your Identity Manager Installation

To determine the best upgrade path, use the worksheets provided in Appendix B, "Assessment Worksheets" to inventory the components of your current Identity Manager installation.

The following sections describe methods for collecting this information:

- Identity Manager Version
- Identity Manager Assessment Tools

Identity Manager Version

Use the Identity Manager Console to verify the version number of your current Identity Manager installation.

1. From the command line, type **`lh console`**.
2. At the prompt, type **`version`**. to display the Identity Manager version number.

Identity Manager Assessment Tools

Identity Manager provides the following utilities to list and record your installation information:

- **`installed` Utility**: Searches the `$WSHOME/bin` directory for manifests and provides version information for releases, patches, service packs, and hotfixes.
- **`inventory` Utility**: Inspects the file system for files that were added to or deleted from the system, using files that are packaged in the release. This utility determines which files were changed based on the manifest that shipped with Identity Manager.

You can access both utilities as follows:

1. Open a command window and change directories to `$WSHOME/bin`.
2. At the prompt, execute the following command:

   ```
   ./lh assessment
   ```

3. At the prompt, type one of the following commands:

   `installed` `[`*option*`] [`*option*`]...`

   `inventory` `[`*option*`] [`*option*`]...`

The following tables describe the options you can use with the `installed` and `inventory` utilities.

Table 2-1 `installed` Utility Options

Option	Function	Description
-h	help	Display usage.
-r	releases	Display only installed releases.
-p	patches	Display only installed patches.
-s	service packs	Display only installed service packs.
-f	hotfixes	Display only installed hotfixes.

NOTE Be sure to record the manifest file names that are associated with all service packs or patches. For example:

```
Identity_Manager_8_0_0_0_20080530.manifest
```

Table 2-2 `inventory` Utility Options

Option	Function	Description
-a	added	Display only added files.
-d	deleted	Display only deleted files.
-h	help	Display usage.
-m	modified	Display only modified files.
-u	unchanged	Display only unchanged files.

Step 3: Document Your Custom Components

Use the worksheets provided in Appendix B, "Assessment Worksheets" to inventory your custom components, including:

- Customized Database Table Definitions
- Custom Filesystem Objects
- Custom Repository Objects

NOTE

- If you are using the Identity Manager IDE or an older version Consolidated Build Environment (CBE), these component customizations should already be part of your baseline. In this case, the CBE baseline serves as your documentation.
- If your current Identity Manager installation has a large amount of custom work, contact Sun Professional Services for assistance with your upgrade.

Customized Database Table Definitions

Identity Manager version 7.1 and version 8.0 made significant changes to Identity Manager's database table definitions.

If you previously modified the database table definitions for the Identity Manager repository, you must carefully decide whether to make the same modifications to the new and updated tables.

Custom Filesystem Objects

You might need to update your customized filesystem objects to enable them to function properly with later Identity Manager releases. List any customized filesystem object names that are in your environment, including:

- Modified JSPs
- Modified `Waveset.properties` File
- Modified `WPMessages.properties` File
- Customized Property Files
- Custom Resource Adapters (and Other Custom Java)
- Modified Stylesheets

Modified JSPs

Recent Identity Manager versions might contain API changes. If you have modified JSPs in your installation, you might have to update them when upgrading. You must update any JSP that was supplied by Identity Manager and changed during a deployment (or a custom JSP that uses Identity Manager APIs) to work with the new JSP structure and API changes for the target release.

NOTE For a detailed description of API changes, see the Identity Manager Release Notes for the release to which you are upgrading.

Use the `inventory -m` command (described on page 16) to identify any JSP modifications made in your deployment.

For more information about JSP customizations, see Appendix A, "Working with Configuration Objects," in the *Sun™ Identity Manager Technical Deployment Overview*.

Modified `Waveset.properties` *File*

Record any changes that you made to the default `Waveset.properties` file.

Modified `WPMessages.properties` *File*

Record any changes that you made to the default `WPMessages.properties` file.

Customized Property Files

Record any changes that you made to other property files on your system.

Custom Resource Adapters (and Other Custom Java)

You might have to recompile your custom resource adapters, depending on the target Identity Manager version. All custom Java that uses Identity Manager APIs (including custom resource adapters) require a recompile during upgrading. Also, consider other Java classes that use the Identity Manager library.

Modified Stylesheets

Record any changes that you made to Identity Manager stylesheets.

Custom Repository Objects

You might have to maintain customized repository objects to enable them to function properly with target Identity Manager releases. Record any customized repository objects that are in your environment, including:

- Modified Forms
- Modified Workflows
- Modified Email Templates
- Custom Repository Schema
- Other Custom Repository Objects

NOTE You can use Identity Manager's SnapShot feature to create a baseline or *snapshot* of the customized repository objects in your deployment, which can be very useful when planning an upgrade. See "Step 5: Take a Snapshot" on page 30 for more information.

Modified Forms

You might have to update customized forms to take advantage of current product enhancements.

Modified Workflows

You might have to update customized workflows to take advantage of current product enhancements.

Modified Email Templates

You might have to export customized email templates to take advantage of current product enhancements.

Custom Repository Schema

Significant schema changes occurred between Identity Manager version 7.0 and version 8.0. If you are upgrading from an earlier version of Identity Manager, you must update your schema.

Other Custom Repository Objects

Record the names of any other custom repository objects that you created or updated. You might have to export these objects from your current installation and then re-import them to the newer version of Identity Manager after upgrading.

- Admin group
- Admin role
- Configuration
- Policy
- Provisioning task
- Remedy configuration
- Resource action
- Resource form
- Role
- Rule
- Task definition
- Task template
- User form

NOTE The SPML 2.0 implementation in Identity Manager has changed in Identity Manager 8.0. In previous releases, the SPML `objectclass` attribute used in SPML messages was mapped directly to the `objectclass` attribute of Identity Manager `User` objects. The `objectclass` attribute is now mapped internally to the `spml2ObjectClass` attribute and is used internally for other purposes.

During the upgrade process the `objectclass` attribute value is automatically renamed for existing users. If your SPML 2.0 configuration contains forms that reference the `objectclass` attribute, you must manually change those references to `spml2ObjectClass`.

Identity Manager does not replace the sample `spml2.xml` configuration file during an upgrade. If you used the `spml2.xml` configuration file as a starting point, be aware that this file contains a form with references to `objectclass` that you must change to `spml2ObjectClass`. Change the `objectclass` attribute in forms (where it is used internally), but *do not* change the `objectclass` attribute in the target schema (where the attribute is exposed externally).

You can use Identity Manager's SnapShot feature to copy the following, specific object types from your system for comparison:

- AdminGroup
- AdminRole
- Configuration
- EmailTemplate
- Policy
- ProvisionTask
- RemedyConfig
- ResourceAction
- Resourceform
- Role
- Rule
- TaskDefinition
- TaskTemplate
- UserForm

For specific instructions, see "Step 5: Take a Snapshot" on page 30.

Task 2: Choose the Target Identity Manager Version

NOTE For the most current description of Identity Manager upgrade paths, see the "Upgrade Paths and Support Policies" section in the *Identity Manager Release Notes*.

In general, you should upgrade to the most recent Identity Manager release that is available during your testing time frame. For example, assume that you are testing now with Identity Manager 7.1.1, as this version was the most current release available when you started your current test cycle. Assume further that the next new Identity Manager release, 7.1.2, is scheduled for July 10th, and that July 15th is the projected start date of your next test cycle. You should plan to upgrade to 7.1.2 when you start your next test cycle.

Be sure that the platform in your Production environment supports the new version of the Identity Manager product. If not, plan to update the platform in each environment *before* you upgrade your Identity Manager application. Reset each target environment to match the Production platform before upgrading that target environment. In general, you must update your platform *as part of the upgrade procedure that you follow in each target environment*.

In cases where both your current Identity Manager product version and the target Identity Manager version support the updated platform, then you can update your platform as a separate change and promote this change all the way to your Production environment before upgrading your Identity Manager application.

The standard upgrade processes that are part of each full-release of Identity Manager generally upgrade an existing installation from any version of the previous major release.

Review the *Identity Manager Release Notes* for the target version of Identity Manager to which you plan to upgrade. The Release Notes document release-specific upgrade considerations. They also contain documentation addenda, bug fixes, and known issues.

Consider your configurations and customizations, and then identify any changes in the Identity Manager product that might affect those configurations and customizations.

Check your current release to see which hotfixes you have installed. Find the bug number associated with each hotfix, and check the Release Notes to confirm that the new, target Identity Manager version contains all of the hot fixes you need.

NOTE Sun's new *patch process* replaces the older hot-fix process. The patch process is cumulative, so you can expect fewer problems with unique fixes. The patch process also makes it easier for you to track a fix by its actual bug number. However, it is still possible that a fix made against an older version might not yet be available in a newer version. Regardless of which process your current version of Identity Manager follows, you must confirm that the new, target Identity Manager version contains all of the bug fixes that you need.

NOTE If you want to upgrade your Identity Manager application more than one level (that is, beyond the next major version from your current version), you must read Appendix A, "Skip-Level Upgrade Considerations." This appendix describes how a skip-level upgrade changes the tasks described in this section.

Task 3: Prepare Your Test Plan

Before proceeding to the next phase of the upgrade, be sure you have prepared a current, comprehensive test plan. The goal of a test plan is to confirm that all your current Identity Manager application functionality remains intact through the upgrade process.

- If you have an existing test plan, read "Review Your Existing Test Plan" on page 24.
- If you have not a prepared test plan yet, create one now using the guidelines described in "Create a Test Plan" on page 24.

Review Your Existing Test Plan

Does your existing test plan address everything you want to test? Is it up-to-date? Is it specific? If not, you must revise your test plan appropriately.

If you are particularly concerned with the performance of a particular set of functions or with items such as the amount of system memory or database space the Identity Manager application consumes, then be sure your test plan also measures these items.

After upgrading the Identity Manager product or after making any significant change to your Identity Manager configurations or customizations be sure to retest your Identity Manager application.

Create a Test Plan

You must create a test plan if you do not already have one prepared for your Identity Manager application. A generic test plan includes:

1. Introduction
    - Description of this document
    - Related documents
    - Schedule and milestones
2. Resource requirements
    - Hardware
    - Software (test tools)

- ❍ Staffing
  - Responsibilities
  - Training

3. Features you are going to test and the test approach
   - ❍ New features testing
   - ❍ Regression testing
4. Features you are not going to test
5. Test deliverables
6. Dependencies and risks
7. Entrance and exit criteria

Task 4: Prepare Your Upgrade Procedure

Before proceeding to the next phase of the upgrade, be sure you have prepared a current, comprehensive upgrade procedure. See "Upgrade Process and Upgrade Procedure" on page 10.

The goal of an upgrade procedure is to specify exactly who does what as you upgrade your Identity Manager application in each environment. You will develop and maintain this upgrade procedure as you upgrade your Identity Manager application in each environment.

- If you have an existing upgrade procedure, read "Review Your Existing Upgrade Procedure" on page 25.
- If you have not yet prepared an upgrade procedure, create one now using the guidelines described in "Create an Upgrade Procedure" on page 26.

Review Your Existing Upgrade Procedure

Does your existing upgrade procedure specify exactly who does what and when as you upgrade your Identity Manager application in each environment? Is it clear how and why the procedure differs in each environment? Is your procedure up-to-date? Does your upgrade procedure contain the same steps for your Test environment and for your QA environment that it does for your Production environment? If not, you must revise your upgrade procedure appropriately.

Are there important considerations that are unique to your Production environment? If so, then your upgrade procedure must rehearse the same steps in your QA environment. See "Special Considerations for Production" on page 57. If the duration of the upgrade procedure in your Production environment is important, then be sure your upgrade procedure says to record the duration of each step in each environment. Upgrading your QA environment should give you a particularly good indication of how long it will take to upgrade your Production environment.

Create an Upgrade Procedure

You must create an upgrade procedure if you have not already prepared one for your Identity Manager application.

An upgrade procedure generally:

- Takes the form of a checklist. Your upgrade procedure may include supporting documentation, but the administrator who performs the upgrade procedure will want a clear, complete, and concise set of instructions.
- Includes most, if not all, of the steps described in "Task 8: Execute Your Upgrade Procedure" on page 46. Your upgrade procedure is generally far more specific, spelling out exactly who must do what in each environment. For example, your procedure must include specific commands and specific parameter values that an administrator must issue in each environment.
- Includes additional steps. For example, you might have to stop and restart external processes if your Identity Manager application integrates with external applications. You might also be required to notify users or systems personnel before taking the Identity Manager application or other affected applications offline.
- Is the same for each target environment. Specific parameter values, such as hostnames and connection information, might vary from environment to environment. The steps in the procedure, however, should be the same in each environment. Even if, for example, there is no one to notify about application downtime in a Test environment or a QA environment, you should rehearse this step in each environment.
- Includes a timetable. Estimate the expected duration for each step, and record the actual duration of each step. The durations that you see in your QA environment are particularly important for predicting the durations that you will see in your Production environment.

Chapter 3

Developing and Testing Your Upgrade

The developing and testing phase of the upgrade consists of the following tasks:

Task 5: Reset Your Development Environment

Task 6: Upgrade Your Development Environment

Task 7: Reset Your Test Environment

Task 8: Execute Your Upgrade Procedure

Task 9: Perform Functional Testing

Task 5: Reset Your Development Environment

You must revert your existing Development environment, or create and set a new Development environment, to the Identity Manager application baseline that corresponds to your Production environment. You must also reset the platform in your Development environment to match the platform in your Production environment. For more information, see “Step 1: Document Your Platform” on page 14.

Use source-control tools to manage your configuration settings, any custom configuration objects, any custom code, test plans, and automated tests.
For more information, see “Source-Control and CBE” on page 8.

If your site's processes allow administrators to change Identity Manager configurations and customizations directly in the Production environment (for example, without updating the baseline version in source-control), then you must compare the current production configurations and customizations to those in the source-control baseline. Identify any changes in the Production environment, apply each change to the Development environment, and re-test as appropriate. Merge these changes into the source-control baseline for your Identity Manager application. If the production changes seem significant and cannot be fully tested in the Development environment, consider promoting the updated Identity Manager baseline to the Test environment and re-testing that baseline before proceeding with the Identity Manager upgrade.

Task 6: Upgrade Your Development Environment

You must perform each of the following steps in your Development environment:

Step 1: Stop Active Sync and Reconciliation

Step 2: Stop the Identity Manager Application

Step 3: Back Up Your Identity Manager Application

Step 4: Remove Hotfixes

Step 5: Take a Snapshot

Step 6: Update Your Platform

Step 7: Upgrade the Identity Manager Product

Step 8: Take Another Snapshot

Step 9: Analyze the Changes

Step 10: Rebuild Any Custom Java

Step 11: Make Necessary Changes in XPRESS

Step 12: Test Your Identity Manager Application in the Development Environment

Step 13: Restart Active Sync and Reconciliation

Step 14: Merge Changes Back into Source-Control

NOTE You must perform some of these steps when upgrading any environment. However, many of these steps are unique to the Development environment because this is the environment where you update the baseline for your Identity Manager application.

Step 1: Stop Active Sync and Reconciliation

Set any Active Sync processes to start manually and, if applicable, disable any scheduled reconciliations until the upgrade is finished and appears to be successful.

➤ **Best Practice**

Step 1 is optional, but performing this step is considered a best practice when upgrading the Production environment.

Also, if you perform Step 1 in your Production environment, make it a standard step when upgrading all of your other environments.

Step 2: Stop the Identity Manager Application

Quiesce your Identity Manager application and make it unavailable to all administrators and end-users.

Step 3: Back Up Your Identity Manager Application

Make a copy of your existing database and Identity Manager file structure.

Backing up the database and file structure enables you to reinstate your working environment, if necessary.

Step 4: Remove Hotfixes

Remove any hotfix class files from your `WEB-INF/classes` directory.

Generally, a hotfix class file works only with the specific version of the Identity Manager product for which that hotfix was delivered.

Step 5: Take a Snapshot

Make a copy of your existing configuration objects. Also, make a copy of other types of objects in the repository; or copy at least a representative sampling of those objects.

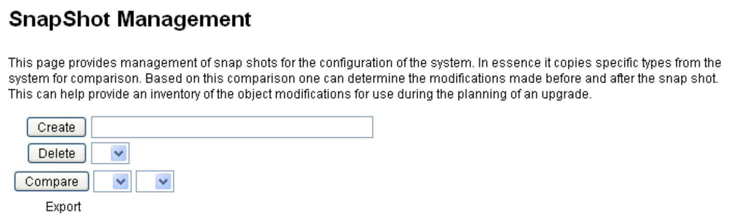
The Identity Manager product upgrade saves the filesystem artifacts that it overlays, such as JSPs, but the upgrade does not preserve "before-images" of every object that it modifies in the repository. Taking a snapshot enables you to detect changes that the Identity Manager product upgrade makes to objects in the repository.

The following instructions describe how to use Identity Manager's SnapShot feature to create a baseline of the customized repository objects in your deployment and how to compare two snapshots to determine what changes have been made to certain system objects before and after upgrade.

NOTE	The SnapShot feature is not intended for detailed, on-going XML diffs — it is only a minimal tool for "first-pass" comparisons.

1. From the Identity Manager Debug page (Figure 3-1), click the SnapShot button to view the SnapShot Management page.

Figure 3-1 SnapShot Management Page



SnapShot Management

This page provides management of snap shots for the configuration of the system. In essence it copies specific types from the system for comparison. Based on this comparison one can determine the modifications made before and after the snap shot. This can help provide an inventory of the object modifications for use during the planning of an upgrade.

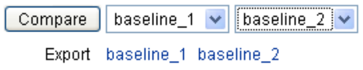
2. Type a name for the snapshot in the Create text box, and then click the Create button.

   When Identity Manager adds the snapshot, the snapshot's name displays in the Compare menu list and to the right of the Export label.

To compare two snapshots:

1. Select the snapshots from each of the two Compare menus (Figure 3-2).

Figure 3-2 SnapShot Management Page



2. Click the Compare button.
   - If no objects were changed, a message indicates no differences were found.
   - If object changes are detected, a message displays the object type and name, and whether the object is different, absent, or present.

     For example, if an object is present in baseline_1, but not present in baseline_2, then the baseline_1 column indicates `Present` and the baseline_2 column indicates `Absent`.

To export a snapshot to a file in XML format, click the snapshot name link.

To delete a snapshot, choose the snapshot name from the Delete menu, and then click the Delete button.

Step 6: Update Your Platform

If the target Identity Manager product version requires changes to your platform, you must makes these changes before upgrading the Identity Manager product.

CAUTION If you are using an Oracle repository, the Identity Manager 8.0 repository DDL uses data types that are not properly handled by older Oracle JDBC drivers. The JDBC drivers in `ojdbc14.jar` do not properly read all of the columns in the log table.

You must upgrade to the `oracle11g_jdbc.jar` drivers for Identity Manager to work properly.

Step 7: Upgrade the Identity Manager Product

To upgrade the Identity Manager product itself, you might be required to:

- Update the Repository Database Tables
- Upgrade the Identity Manager Product
- Upgrade Gateway and Password Sync

Update the Repository Database Tables

Most major releases and some minor releases of Identity Manager include database table changes. Consequently, you might have to modify the sample SQL scripts for your environment.

You must also update the database tables if you made any of the following modifications:

- Changed the database instance name
- Changed the name of the database account that owns the database tables
- Separated the owner of the database tables from the database account used to connect to the database
- Made more-advanced DBA changes, such as configuring specific table spaces and growth characteristics for different sets of tables and indexes

You must remember any changes you make to the sample SQL scripts for each Identity Manager version and use source-control to manage these changes. In the future, you will have to make similar changes to the sample SQL scripts for subsequent Identity Manager versions.

Upgrade the Identity Manager Product

You can use either of the following methods to upgrade the Identity Manager product:

- Use Identity Manager's installer program as described on page 34.
- Use Identity Manager's manual upgrade process as described on page 35.

Both methods produce the same results.

NOTE In some environments, including on HP-UX, you might prefer using the manual upgrade procedure. For example,

- If you want to fully automate the upgrade as part of a repeatable upgrade procedure
- If you have restricted access to your Production environment or cannot start the console

Upgrading the Identity Manager product might modify objects in the Identity Manager repository and in some filesystem artifacts such as JSPs, Identity Manager product JARs, and third-party JARs.

When upgrading the Identity Manager product, be aware of the following:

- If you copy files from the installation media to your own location, you must put the `idm.war` and `install.class` files in the same directory.
- Use only one Identity Manager server to import `update.xml`, and have only one Identity Manager server running during the upgrade.

  If you start any other Identity Manager servers during the upgrade, you must stop and restart those servers before making them available.
- If your application server is installed on a UNIX machine, change directories to the `$WSHOME/bin` directory and run the following command to allow the scripts in this directory to be executed.

  ```
  chmod -R +x *
  ```
- For UNIX environments, be sure you have an `install` directory in one of the following locations and that you can write to that directory:
  - **For Linux/HP-UX**: `/var/opt/sun/install`
  - **For Solaris**: `/var/sadm/install`
- Previously installed hotfixes are archived to the `$WSHOME/patches/`*`HotfixName`* directory.

Troubleshooting Upgrade

If you encounter problems during the upgrade, check the upgrade log files located in the `$WSHOME/patches/logs` directory. The file names for the logs are based on a timestamp and the stage of the upgrade.

Using the Identity Manager Installer

Use the following steps to upgrade your Development environment using the Identity Manager installation and upgrade program:

1. Use one of the following methods to start the installer:
   - To use the GUI installer, run the `install.bat` (for Windows) or `install` (for UNIX).

     The installer displays the Welcome panel.
   - To activate the installer in `nodisplay` mode, change to the directory where the software is located, and enter the following command:

     ```
     install -nodisplay
     ```

     The installer displays the Welcome text, and then presents a list of questions to gather installation information in the same order as the GUI installer.

   NOTE
   - If no display is present, the installer defaults to the `nodisplay` option.
   - The installer does not install an older version of the software over a newer version. In this situation, an error message displays and the installer exits.

2. On the Welcome panel, click Next.
3. On the Install or Upgrade? panel, select Upgrade, and then click Next.
4. On the Select Installation Directory panel, select the directory where the earlier Identity Manager version is located and click Next.

   The installer displays progress bars for the pre- and post-upgrade processes and then proceeds to Installation Summary panel.
5. For detailed information about the installation, click Details, view the log file, and click Close to exit the installer.
6. Remove all of the compiled Identity Manager files from the work directory of the application server.

7. If you are running Gateway on a remote system, upgrade it by using the following steps.

   a. Log in to the Windows system, and change to the directory where Gateway is installed.

   b. Stop the Gateway service by running the command:

      ```
      gateway -k
      ```

   c. If using Windows 2000 or later, exit all instances of the Services MMC plug-in.

   d. Use the following command to remove the Gateway service:

      ```
      gateway -r
      ```

   e. Back up and delete the existing Gateway files.

   f. Extract the new Gateway files.

      If you are installing the newly upgraded Gateway on a system that is not the Identity Manager server, then copy the `gateway.zip` file from the Identity Manager Installation CD.

   g. Unpack the `gateway.zip` file into the directory where Gateway was installed.

   h. Run the following command to install the Gateway service:

      ```
      gateway -i
      ```

   i. Run the following command to start the Gateway service:

      ```
      gateway -s
      ```

Upgrading Manually

In some environments, you might want to perform the upgrade steps manually instead of using the Identity Manager installation and upgrade program.

NOTE

- Be sure you set the `JAVA_HOME` environment variable.
- Make sure that the `bin` directory in the `JAVA_HOME` directory is in your path.
- Any previously-installed hotfixes will be archived to the `$WSHOME/patches/`*HotfixName* directory.

The instructions in this section are based on installing Identity Manager on a Tomcat application server. Depending on your application server, you might have to use slightly different commands.

NOTE Refer to the appropriate chapter in *Sun Java™ System Identity Manager Installation* for application server-specific instructions.

➤ To Install on a Windows Platform

Use the following steps to upgrade Identity Manager manually on a supported Windows platform:

1. Stop the application server and Sun Identity Manager Gateway.
2. Update the Identity Manager database.
3. Enter the following commands to set your environment:

```
set ISPATH=Path to install software
set WSHOME=Path to Identity Manager Installation OR Staging Directory
set TEMP=Path to Temporary Directory
```

NOTE If you have a space in the path to the Identity Manager installation directory, you must specify the `WSHOME` environment variable without double quotes (`"`), as shown in the following example.

Do not use trailing slashes (`\`) when specifying the path even if the path contains no spaces.

```
set WSHOME=c:\Program Files\Apache Group\Tomcat 6.0\idm
```

or

```
set WSHOME=c:\Progra~1\Apache~1\Tomcat~1\idm
```

The following path will not work:

```
set WSHOME="c:\Program Files\Apache Group\Tomcat 6.0\idm”
```

4. Run pre-process:

```
mkdir %TEMP%
cd /d %TEMP%
jar -xvf %ISPATH%\IDM.WAR \
WEB-INF\lib\idm.jar WEB-INF\lib\idmcommon.jar
set TMPLIBPTH=%TEMP%\WEB-INF\lib
set CLASSPATH=%TMPLIBPTH%\idm.jar;\
%TMPLIBPTH%\idmcommon.jar;
java -classpath %CLASSPATH% -Dwaveset.home=%WSHOME% \
   com.waveset.install.UpgradePreProcess
```

5. Install software:

```
cd %WSHOME%
jar -xvf %ISPATH%\IDM.WAR
```

6. Run post-process:

```
java -classpath %CLASSPATH% -Dwaveset.home=%WSHOME%
  com.waveset.install.UpgradePostProcess
```

NOTE The installer supports upgrading installations that have renamed, deleted, or disabled the default Configurator account.

The installer prompts you for user name and password to import the `update.xml` during the upgrade post process. If the user or password is entered incorrectly, you will be prompted (up to three times) to enter the correct password. The error will be displayed in the text box behind it.

For manual installation you must provide the `-U` *username* `-P` *password* flags to pass the credentials to the `UpgradePostProcess` procedure.

7. If you installed into a staging directory, create a `.war` file for deployment to your application server.

8. Remove the Identity Manager files from the application server work directory.

➤ To Install on a Unix Platform

Use the following steps to upgrade Identity Manager manually on a supported Unix platform:

1. Stop the application server and Sun Identity Manager Gateway.
2. Update the Identity Manager database.
3. Enter the following commands to set your environment:

   export ISPATH=*Path to Install Software*
   export WSHOME=*Path to Identity Manager Installation* OR *Staging Directory*
   export TEMP=*Path to Temporary Directory*

4. Run pre-process:

```
mkdir $TEMP
cd $TEMP
jar -xvf $ISPATH/idm.war \
WEB-INF/lib/idm.jar WEB-INF/lib/idmcommon.jar
CLASSPATH=$TEMP/WEB-INF/lib/idm.jar:\
$TEMP/WEB-INF/lib/idmcommon.jar:
java -classpath $CLASSPATH -Dwaveset.home=$WSHOME \
com.waveset.install.UpgradePreProcess
```

5. Install software:

```
cd $WSHOME
jar -xvf $ISPATH/idm.war
```

6. Run post-process:

```
java -classpath $CLASSPATH -Dwaveset.home=$WSHOME
  com.waveset.install.UpgradePostProcess
```

NOTE The installer supports upgrading installations that have renamed, deleted, or disabled the default Configurator account.

The installer prompts you for user name and password to import the `update.xml` during the upgrade post process. If the user or password is entered incorrectly, you will be prompted (up to three times) to enter the correct password. The error will be displayed in the text box behind it.

For manual installation you must provide the `-U` *username* `-P` *password* flags to pass the credentials to the `UpgradePostProcess` procedure.

7. Change directory to `$WSHOME/bin/solaris` or `$WSHOME/bin/linux`, and then set permissions on the files in the directory so that they are executable.
8. If you installed into a staging directory, create a `.war` file for deployment to your application server.
9. Remove the Identity Manager files from the application server work directory.

Upgrade Gateway and Password Sync

Upgrade every Sun Identity Manager Gateway and Password Sync installation in your environment.

- To upgrade the Gateway, see Step 7 on page 35 for instructions.
- To upgrade PasswordSync, you must uninstall each PasswordSync installation in your environment and reboot. Replace each installation with the new PasswordSync version and reboot. The two reboots are necessary due to the way the Windows Security Service loads the PasswordSync `dll`.

  See "Installing PasswordSync on Windows" in *Identity Manager Administration* for installation instructions.

Step 8: Take Another Snapshot

After successfully upgrading the Identity Manager product, make a copy of the existing configuration objects. Also, make a copy of other objects types in the repository; or copy at least a representative sampling of those objects.

The Identity Manager product upgrade does not record the changes it makes to repository objects. If you compare this snapshot to the snapshot that you took before the upgrade, you can easily detect any changes made to repository objects during the upgrade.

Step 9: Analyze the Changes

You must analyze the changes made by the Identity Manager product upgrade and update your configurations and customizations accordingly. For example,

- If you modified any JSP files or stylesheets, you must merge these changes into the new JSP files or stylesheets.
- If your Identity Manager application baseline includes Identity Manager product JARs and third-party JARs, you might have to update these JARs in the baseline. Your baseline should also include SQL scripts used to create or update your database tables.
- If you modified any of the default Identity Manager objects (such as the Default User Form), the upgrade process moves those objects into the `savedObjects` directory. To facilitate future upgrades, rename the modified objects with a custom name and reference that name in the `SystemConfiguration` object.
- If you extracted `WPMessages.properties` to the `/config` directory and customized any of the messages, you must extract and reapply these customizations.

In particular, you must carefully analyze changes made to repository objects during the Identity Manager product upgrade. For example, if the Identity Manager product upgrade modified

- Configuration objects that are in your source-control baseline, you must merge these changes into your configuration baseline. For more information, see "Step 14: Merge Changes Back into Source-Control" on page 44.
- Configuration objects that are not currently in your baseline, you must add these objects to your application baseline. If you do not add these configuration objects to your application baseline, then you must make other plans to incorporate these changes, such as including the appropriate objects or commands within the subset of `update.xml` that your upgrade procedure imports in each environment.

> ➤ **Best Practice**
>
> You might decide that you can safely ignore these object changes, but in most cases it is considered best practice to add these configuration objects to your baseline.

- Objects in the repository that are *not* configuration objects, and these objects should *not* become part of your source-control baseline. For example, Identity Manager's `update.xml` might refresh `TaskInstance` objects, `User` objects, `Account` objects, or `Entitlement` objects.

  Identity Manager Engineering generally avoids updating these object types because there can be so many instances of each type, but in some cases changes are necessary or justified. In such cases, include executing an appropriate subset of Identity Manager's `update.xml` in your baseline and in your upgrade process. Use this `update.xml` subset to update repository objects that are not part of your baseline.

After upgrading, restore any customized files and objects.

Restoring Customized Files

During upgrade, Identity Manager automatically copies all customized files, such as JSPs and HTML, into the following directory.

```
$WSHOME/patches/Sun_Java_System_Identity_Manager_Version_Date_/savedFiles
```

The following table describes the files in this directory.

Table 3-1 `savedFiles` Directory File Structure

File Name	Description
`changedFileList`	File containing a list of all saved customized files. This file also contains a list of files (installed with your older version of Identity Manager), that will be overwritten when files of the same name are installed during upgrade.
`notRestoredFileList`	File containing a list of all customized files that are *not* restored during the upgrade process.
`notInstalledFileList`	File containing a list of newer version files that are not installed during the upgrade process.

The upgrade might add some files that were also installed with your original Identity Manager installation. Before overwriting the older files, Identity Manager automatically saves them in the `savedFiles` directory. See the `changedFileList` file for a list of these files.

Identity Manager automatically restores most of the files listed in `changedFileList` during the upgrade process, but does not restore all of them. See the `notRestoredFileList` for a list of these files. When restoring customized files, Identity Manager overwrites the newer version of the files that were installed during upgrade.

You might have to manually restore some of your file customizations. Review the `notRestoredFileList` file to see a list of the files that were *not* restored during upgrade. If you must manually restore any customized files, edit the new file that was installed during upgrade to incorporate your customizations, and then save the newly edited file.

Restoring Customized Objects

If you have configured your form and process mappings in system configuration, you will not have to restore those object customizations after the upgrade. If you have customized objects that are not listed in the system configuration, then you must manually restore these objects by importing the XML for these objects.

As a safety measure, Identity Manager automatically saves many of the commonly customized objects to files when you import `update.xml`. These files are saved to subdirectories in the `WEB-INF/savedObjects` directory. These subdirectories are named with timestamp of the time at which the import was performed.

Importing `update.xml` can create up to three subdirectories in the `savedObjects` directory. You can manually import the object XML files to restore object customizations.

Step 10: Rebuild Any Custom Java

You must rebuild all of your custom Java classes against the new product libraries For example, you must rebuild any new JAR files, new JDK, or application server libraries.

If recompiling produces deprecation warnings, analyze the deprecation messages, and read the *Identity Manager Release Notes*, to determine whether you can resolve the deprecation message immediately. If you cannot resolve the deprecation issue immediately, add an item to your project plan to resolve the deprecation message in the future.

NOTE	Identity Manager does not support deprecated APIs indefinitely. Deprecated classes and methods are generally removed in the next major product release.

Step 11: Make Necessary Changes in XPRESS

Make any forms, rules, and workflows changes in XPRESS.

The forms, rules, and workflows supplied in new Identity Manager product versions are generally backward-compatible with older forms, rules, and workflows. The most common type of change required is to change invocations of Identity Manager Workflow Services or Form Utility methods.

NOTE For information about release-specific changes to Workflow Services or Form Utility methods, see the Identity Manager Release Notes for the release to which you are upgrading.

Step 12: Test Your Identity Manager Application in the Development Environment

Restart the application server and test your Identity Manager application at least minimally to confirm that at least the basic functions are working as expected.

You must redeploy your web applications after upgrading Identity Manager because most application servers cache the `web.xml` file.

For example, if you are using the Sun Java™ System Application Server, you would perform the following steps to redeploy a web application after upgrading Identity Manager.

1. Log in to the Sun Java System Application Server Administrator interface.
2. Select Applications > Web Applications from the menu bar.
3. Locate your web application and click the Redeploy link.
4. Enable the button next to the Local Packaged File or Directory That is Accessible From the Application Server option.
5. Click the Browse Folders button and select the top-level folder for your installation. For example

   ```
   C:\Sun\AppServer\domains\domain1\applications\
   j2ee-modules\idm
   ```

6. Click OK.
7. Restart the application server.

Step 13: Restart Active Sync and Reconciliation

After successfully upgrading, you must restore the original settings for any Active Sync processes and re-enable any scheduled reconciliations (if applicable).

➤ **Best Practice**

Step 13 is optional, but performing this step is considered a best practice when upgrading the Production environment.

Also, if you perform Step 13 in your Production environment, make it a standard step when upgrading all of your other environments.

Step 14: Merge Changes Back into Source-Control

NOTE Merging changes back into source-control is specifically listed here as a separate step to highlight its importance. In actual practice, you can merge changes back into source-control as you perform Steps 9 through 12.

When merging changes back into source-control, you must:

- Verify that all existing customizations are tagged and stored in the version control system.
- Check in all new customizations after completing the test upgrade cycle, including modified:
    - Database table scripts
    - Repository objects
    - Filesystem objects, such as JSPs and images

Task 7: Reset Your Test Environment

To perform controlled testing, you must reset your Test environment so that it corresponds to your Production environment as closely as possible.

Step 1: Reset Your Platform to Match Production

To reset your Test environment, ensure that:

- The platform in the Test environment matches your current Production environment. The platform includes the application server, Repository DBMS, and JDK version. See "Step 1: Document Your Platform" on page 14.
- The Identity Manager application image in your Test environment corresponds to the application baseline for your current Production environment.
- The database table definitions in the Test environment match those in the Production environment.
- Resources and other integrated applications match those in the Production environment.

  If real test resources do not exist, you can create simulated resources for the functional test.

Every time you promote an image of your Identity Manager application from the Development environment, you must test your cumulative upgrade procedure. If the upgrade procedure appears to be successful, execute your test plan.

Step 2: Set Up for Functional Testing

To prepare for functional testing, you must create a Test environment that supports controlled testing of your Identity Manager application.

You might want to simulate some aspects of the Production environment, but the primary goal is to verify that the application works as expected. Achieving this goal might require that you to load controlled datasets rather than perfectly realistic ones.

Load test data into your database tables that supports execution of the test cases in your test plan. Ideally, the database tables would also contain data similar to the data in your Production environment.

Task 8: Execute Your Upgrade Procedure

Upgrading a Test environment requires only a subset of the steps you performed when upgrading your Development environment. For example, you do not have to detect changes or update source-control. The updated baseline for your Identity Manager application already contains those changes.

Before upgrading any targeted environments, you must generate an image of your Identity Manager application that is appropriate for that environment. The baseline, and therefore the image, contains

- SQL scripts that update the database tables, filesystem objects, repository objects
- An appropriate subset of `update.xml` to update repository objects that are not in your baseline

Step 1: Stop Active Sync and Reconciliation

Set any Active Sync processes to start manually and, if applicable, disable any scheduled reconciliations until the upgrade is complete and appears to be successful.

➤ **Best Practice**

Step 1 is optional, but performing this step is considered a best practice when upgrading the Production environment.

Also, if you perform Step 1 in your Production environment, make it a standard step when upgrading in all of your other environments.

Step 2: Stop the Identity Manager Application

Quiesce your Identity Manager application and make it unavailable to all administrators and end-users.

Step 3: Back Up Your Identity Manager Application

Make a copy of your existing database and Identity Manager file structure.

Backing up the database and file structure enables you to reinstate your working environment, if necessary.

NOTE	Always back up the Identity Manager database and file system before applying any Identity Manager patches, service packs, or hotfixes and before going through any major upgrades.

You can use third-party back-up software or a back-up utility supplied with your system to back up the Identity Manager file system. To back up your database, see your database documentation for recommended back-up procedures.

When you are ready to create a backup, you must

1. Shutdown or idle Identity Manager.
2. Use your back-up utilities to back up your database and the file system where you installed Identity Manager.

Step 4: Remove Hotfixes

Remove any hotfix class files from your `WEB-INF/classes` directory.

Hotfix class files generally work only with the specific version of the Identity Manager product for which the hotfix was delivered.

Step 5: Change `TaskDefinition` Objects

You might find it necessary to upgrade a Production environment that contains executing `TaskInstances`. Unfortunately, upgrading an Identity Manager `TaskDefinition` object in the repository can corrupt executing task instances that depend on the `TaskDefinition` object. This possibility is a particularly important consideration in a Production environment where people are depending on those tasks to complete correctly and to perform their business functions.

Although it is easiest to have users complete their tasks or terminate still-executing tasks prior to upgrade, these options are not always feasible.

If your Production environment might contain executing task instances when you upgrade, be sure your upgrade procedure describes how to address these instances.

➤ **Best Practice**

Rename `TaskDefinition` objects when upgrading in each environment. Use the following process to upgrade `TaskDefinition` objects in your Production environment:

1. From the Identity Manager console, rename the current `TaskDefinition` to include a timestamp.

   For example, rename `Create User` to `Create User 20030701`.

   You must rename the `TaskDefinition` object to prevent any problems with existing `Create User` tasks that might be in a suspended state in Identity Manager. Renaming the `TaskDefinition` object allows the existing `TaskDefinition` to keep its unique ID, which is referenced inside suspended tasks.

2. Load the new `TaskDefinition`.

CAUTION Problems might occur if you change activities or actions.

You may not modify any `TaskDefinitions` that correspond to live `TaskInstances`. Identity Manager does not allow you to make these modifications.

Step 6: Update Your Platform

If the target Identity Manager product version requires platform changes, you must make these changes before upgrading the Identity Manager product.

Step 7: Upgrade Your Identity Manager Application

To upgrade your Identity Manager application, you might be required to:

- Update Your Database Table Definitions
- Promote the Identity Manager Application
- Import a Subset of `update.xml`
- Upgrade Your Gateway and Password Sync Components

NOTE **About Data Sources**

If you use a JDBC data source defined in your application server as your Identity Manager repository location, be aware that this data source might not work outside the application server. In other words, a JDBC data source provided by an application server might be available for use only by web applications that run in that container.

The Identity Manager product upgrade process runs outside the application server, just like the Identity Manager console. Therefore, in each environment where Identity Manager normally uses a DataSource, your upgrade procedure might need to include steps to switch to a JDBC DriverManager connection.

You can temporarily replace the `ServerRepository.xml` file that specifies a data source with another `ServerRepository.xml` file that specifies a JDBC DriverManager connection. Restore the original `ServerRepository.xml` file as a subsequent step in your upgrade procedure.

Alternatively, you can expand the Identity Manager application `WAR` file onto the filesystem, specify `WSHOME` as the filesystem location, and use this “side” environment to perform a manual upgrade process or to perform any step that requires a console, such as importing a subset of `update.xml` or renaming `TaskDefinition` objects.

If additional set up is required for your custom integrations in each environment, perform the additional set up as part of this step.

Update Your Database Table Definitions

Verify that your Identity Manager application image includes any SQL scripts needed to update your database table definitions, and that these SQL scripts have been modified to fit your environment.

If your image does not include these SQL scripts, ensure your upgrade procedure specifically describes the modifications required for each environment.

Promote the Identity Manager Application

Promote the Identity Manager application image into your Test environment. Your application image must include the target Identity Manager product version, your updated configuration, and your customizations.

Import a Subset of `update.xml`

You must import the `update.xml` file to update the repository objects that are not managed as part of your Identity Manager application baseline.

➤ **Best Practice**

Use only one Identity Manager server to import `update.xml` and have only one Identity Manager server running during the upgrade.

If you start any other Identity Manager servers during the upgrade process, you must stop and restart those servers before making them available again.

Upgrade Your Gateway and Password Sync Components

Upgrade every Sun Identity Manager Gateway or Password Sync installation in your environment.

Step 8: Test Your Identity Manager Application

Restart the application server and test your Identity Manager application at least minimally to verify that the basic functions are working as expected.

You must redeploy your web applications after upgrading Identity Manager because most application servers cache the `web.xml` file.

For example, if you are using the Sun Java™ System Application Server, you would perform the following steps to redeploy a web application after upgrading Identity Manager.

1. Log in to the Sun Java System Application Server Administrator interface.
2. Select Applications > Web Applications from the menu bar.
3. Locate your web application and click the Redeploy link.
4. Enable the button next to the Local Packaged File or Directory That is Accessible From the Application Server option.
5. Click the Browse Folders button and select the top-level folder for your installation. For example

   ```
   C:\Sun\AppServer\domains\domain1\applications\
   j2ee-modules\idm
   ```

6. Click OK.

Restart the application server.

Step 9: Restart Active Sync and Reconciliation

After successfully completing the upgrade, restore the original settings for any Active Sync processes and for any scheduled reconciliations.

➤ **Best Practice**

Step 8 is optional, but performing this step is considered a best practice when upgrading the Production environment.

Also, if you perform Step 8 in your Production environment, make it a standard step when upgrading all of your other environments.

Step 10: Restart the Identity Manager Application

Restart the Identity Manager application to make the application available again to administrators and end-users.

Task 9: Perform Functional Testing

Testing in the Test environment is crucial before you deploy the Development upgrade image into your Production environment.

To test your Test environment after upgrading,

1. Execute your complete test plan, including any automated tests.
2. Fix any problems and incorporate the fixes into the source-control baseline in your Development environment.
3. Repeat the process of resetting your Test environment, upgrading your Test environment, and retesting your Identity Manager application.

Chapter 4

User Acceptance Testing

The User Acceptance Testing phase consists of the following tasks, which are very similar to Tasks 6, 7, and 8 described in Chapter 3:

Task 10: Reset Your QA Environment

Task 11: Upgrade Your QA Environment

Task 12: Perform User Acceptance Testing

The document describes user acceptance testing as a separate upgrade phase for several reasons:

- The QA environment must replicate the Production environment as closely as possible, which allows you to test the upgrade procedure against realistic data values and against a realistic volume of data. For more information, see "Task 10: Reset Your QA Environment" on page 54 and "Task 11: Upgrade Your QA Environment" on page 55.
- A skip-level upgrade greatly affects how you develop and test your upgrade, but does not affect user acceptance testing. For more information, see Appendix A, "Skip-Level Upgrade Considerations" on page 59.
- Most importantly, to highlight the importance of the user community. You use this testing phase to verify that your Identity Manager application meets the needs of the people who use or who are intended to use the application. For more information, see "Task 12: Perform User Acceptance Testing" on page 55.

Task 10: Reset Your QA Environment

Reset your to your QA environment to mimic your Production environment as closely as possible, being sure to use the same hardware and software versions used in the Production environment.

Specifically, you want to replicate your Production environment's

- Platform, including
  - Application server
  - Database server
  - Gateway server
  - Web Server (optional component)
  - Common client machine with the corporate image and same browser
- Identity Manager product version
- Identity Manager application configurations and customizations
- Database tables and actual data, including data values and the volume of data found in the Production environment
- Resources and other integrated applications

  Configure resources and other integrated applications in the QA environment to have the same general configuration as those used in the Production environment. For example, replicate active directory domains, organizations, groups, numbers, and the format of users from Production to the QA environment.

  Also, be sure to configure integrated applications the same way in both environments and that these applications contain a copy of recent production data or very similar data.

Task 11: Upgrade Your QA Environment

The process for upgrading your QA environment is very similar to the process described in "Task 8: Execute Your Upgrade Procedure."

Other than a few different parameter values, such as host names and connection information, your procedure for updating your QA environment must contain exactly the same steps as your procedure for upgrading the Test environment.

Because your QA environment is supposed to replicate the Production environment as closely as possible, you must monitor the upgrade closely for any errors caused by data values, data volumes, or platform considerations that were not fully tested in the Test environment.

To help you plan the final phase of your upgrade, upgrading your Production environment, carefully measure how long it takes you to execute various steps in your upgrade procedure.

Task 12: Perform User Acceptance Testing

Performing user acceptance testing is comparable to performing "Task 9: Perform Functional Testing" on page 52. In fact, you might choose to execute your test plan or a subset of the plan in this simulated Production environment.

The main difference between these two tasks is that for user acceptance testing, you must arrange for the people who actually use your Identity Manager application to test it with realistic data.

Though it might be difficult to schedule user participation with people distributed across your organization, user acceptance testing is usually quite valuable. Giving users an advanced look at the next version of your Identity Manager application generally helps maintain productivity and encourages adoption. Conducting user acceptance testing also demonstrates that the deploying organization is proactive and responsive to the needs of the people who are using the application.

User acceptance testing often uncovers problems and clarifies requirements for your Identity Manager application. The users might also find issues that are specific to aspects of your platform that were not tested in other environments. Although most developers prefer to find problems in earlier phases, finding problems during user acceptance testing is still much better than finding problems after you upgrade the Production environment.

Even if you ultimately decide to go into production with issues or limitations, you will know about these issues or limitations ahead of time. Key users are then prepared to communicate the problems and any workarounds to other users of the application.

As with Task 8, if you have to fix any problems, you must

1. Incorporate the fixes back into your Development environment's source-control baseline.
2. Reset your Test environment.
3. Upgrade your Test environment.
4. Retest your Identity Manager application.
5. Generally, repeat User Acceptance Testing. You might decide that incremental testing is sufficient.

Chapter 5

Upgrading Your Production Environment

Although you perform only one task to upgrade your Production environment, and the process for upgrading Production is very similar to the processes described in "Task 8: Execute Your Upgrade Procedure" and "Task 11: Upgrade Your QA Environment," upgrading Production is specifically described separately to highlight its importance.

Special Considerations for Production

Your upgrade procedure probably includes steps that are important only when you are upgrading your Production environment. For example, your upgrade procedure might include steps to

- Schedule an outage for your Identity Manager application
- Schedule database administrator support
- Notify users before taking the application offline
- Shut down specific resources, processes, or applications that are used only in your Production environment

Upgrading your Production environment might also pose special considerations that require advanced techniques. For example, to minimize the downtime of your Identity Manager application, you might want to upgrade a separate server and a separate database in a parallel Production instance. After upgrading the parallel instance, you might want to make the original Production instance unavailable, synchronize from the actual Production system to the parallel instance any data that changed during the time that the parallel instance was being upgraded, and then switch over to the parallel Production instance.

Advanced techniques such as these are beyond the scope of this document. If you have any special considerations or technical questions related to upgrading your Production environment, contact your Sun Professional Services to request assistance.

Task 13: Upgrade Your Production Environment

The procedure for updating the Production environment must contain the same steps as your procedure for updating the QA environment. Some specific parameter values, such as host names and connection information can be different.

If there is any significant difference between the procedure for upgrading the Test environment and the procedure for upgrading the Production environment, then document the specific reasons for this variance. For example, it might not be feasible to simulate certain aspects of the Production environment in the QA environment. By documenting this variance, you allow the people involved in your Upgrade Project to make an informed judgment as to the risk posed by this variance.

Appendix A

Skip-Level Upgrade Considerations

This appendix describes the special considerations related to performing a *skip-level* upgrade.

The topics are organized as follows:

- Overview
- Phases of a Skip-Level Upgrade
- Task 2: Choose the Target Identity Manager Version (Enhanced)
- Task 5: Upgrade Your Development Environment (Repeated)

Overview

You develop a *skip-level* (or *multi-hop*) upgrade to update your Production environment to a version of the Identity Manager product that is beyond the next major release from your current version. For example, if you are currently using Identity Manager version 6.0 and want to upgrade to Identity Manager version 8.0, while upgrading your Production environment only once, this upgrade path would require a skip-level upgrade.

Upgrading several versions normally requires a *series of upgrades*. However, many customers want to minimize the number of upgrades in the Production environment, which in turn minimizes the *downtime* of your Identity Manager application, minimizes the cost of *retesting* the Identity Manager application, and minimizes the cost of *retraining* the people using the Identity Manager application.

Developing a single upgrade procedure to update a target Identity Manager version is technically possible and some customers find it feasible. The key point to understand is that, even though you are performing a skip-level upgrade in your Production environment, you still must perform each *hop* in your Development environment. For example, you must upgrade from Identity Manager version 6.0 to version 7.1 to version 8.0 in your Development environment. After each hop, you build up set of artifacts that is *cumulative*.

The most common approach for a skip-level upgrade is to update your application baseline with configurations and customizations that have been updated to work with each Identity Manager version on the path to your target version of Identity Manager. Your baseline also includes a cumulative script to update databases and a cumulative subset of `update.xml`. The net effect is that you develop a single upgrade procedure that makes changes that are *equivalent to the changes that performing the series of upgrades would have done*.

Performing a skip-level upgrade is more complex than a standard or *single-hop* upgrade. Skip-level upgrades require more technical insight into the mechanisms used by Identity Manager's product upgrade. A skip-level upgrade also requires closer analysis of the upgrade content for each Identity Manager product version in the upgrade path. Closer analysis allows you to produce artifacts that are properly cumulative and yet minimal. For example, if you simply combine all of the database table upgrade scripts into one script or if you combine the subsets of `update.xml` from each step into one subset, then your upgrade might perform a great deal of redundant processing.

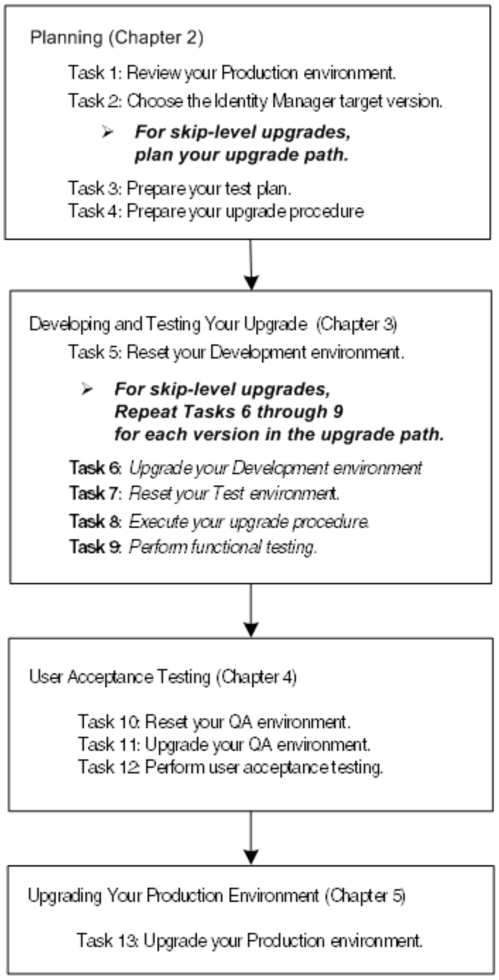
This appendix describes special considerations for a skip-level upgrade. If you are uncertain how to proceed or if these considerations seem unclear to you, then contact Sun Professional Services to request assistance with planning your upgrade.

Phases of a Skip-Level Upgrade

At a high-level, the steps you perform for a skip-level upgrade are the same as those performed for a regular upgrade. However, some these steps are enhanced and some steps are repeated for a skip-level upgrade.

The following figure illustrates the skip-level upgrade process.

Figure 5-1 Skip-Level Upgrade Phases


Planning (Chapter 2)
Task 1: Review your Production environment.
Task 2: Choose the Identity Manager target version.
For skip-level upgrades, plan your upgrade path.
Task 3: Prepare your test plan.
Task 4: Prepare your upgrade procedure
Developing and Testing Your Upgrade (Chapter 3)
Task 5: Reset your Development environment.
For skip-level upgrades, Repeat Tasks 6 through 9 for each version in the upgrade path.
Task 6: Upgrade your Development environment
Task 7: Reset your Test environment.
Task 8: Execute your upgrade procedure.
Task 9: Perform functional testing.
User Acceptance Testing (Chapter 4)
Task 10: Reset your QA environment.
Task 11: Upgrade your QA environment.
Task 12: Perform user acceptance testing.
Upgrading Your Production Environment (Chapter 5)
Task 13: Upgrade your Production environment.

For planning purposes, the first difference between a regular upgrade and a skip-level upgrade is in how you perform "Task 2: Choose the Target Identity Manager Version." When choosing the target Identity Manager version, you must plan your upgrade path.

The next, and biggest, difference between the two upgrade processes is that you must perform Task 6: Upgrade Your Development Environment through Task 9: Perform Functional Testing *for each stop in the upgrade path*. You must upgrade your Development environment and your application baseline for the Identity Manager product version *at each stop in the upgrade path*. You probably also want to retest after each stop in the upgrade path, which requires resetting your Test environment and promoting your Identity Manager application to the Test environment after each stop in the upgrade path.

➤ **Best Practice**

You might think retesting after each stop on the upgrade path is unnecessary, but testing the upgrade process and testing the Identity Manager application at each stop on the upgrade path minimizes your risk because you identify any problems quickly. Working with a smaller set of changes makes it much easier to isolate the cause of any problems.

Task 2: Choose the Target Identity Manager Version (Enhanced)

When performing a skip-level upgrade, you must plan your upgrade path after deciding which Identity Manager version is your target.

NOTE For information about Identity Manager upgrade paths, read the "Upgrade Paths and Support Policies" section in the *Identity Manager Release Notes*.

The standard upgrade processes supplied with each full Identity Manager release generally upgrade an existing installation from any version of the previous major release.

Remember that each stop in the upgrade path requires a different version of the Identity Manager product. To plan your skip-level upgrade, you must read the Release Notes provided for the Identity Manager product version *at each stop in the upgrade path*.

For example, if you are upgrading Identity Manager from version 6.0 to version 8.0, you must read the Identity Manager Release Notes for

- Version 6.0 and all 6.*x* service packs
- Version 7.0 and all 7.*x* service packs
- Version 7.1 and all 7.*x* patches
- Version 8.0

Task 5: Upgrade Your Development Environment (Repeated)

When performing a skip-level upgrade, you must repeat Task 5 once *for each stop in the upgrade path*.

Perform Steps 1 – 15 as described in "Task 6: Upgrade Your Development Environment" on page 28 *for each Identity Manager product version* to which you must upgrade to reach your Identity Manager target version:

NOTE Of these steps, only "Step 9: Analyze the Changes" must be significantly enhanced for a skip-level upgrade. See the following section, "Step 9: Analyze the Changes for a Skip-Level Upgrade," for the enhanced instructions.

Step 9: Analyze the Changes for a Skip-Level Upgrade

You must analyze the changes made by the Identity Manager product upgrade.

As you iteratively upgrade your Development environment, you iteratively update the baseline for your Identity Manager application, including

- Configuration objects
- JSP files
- Identity Manager product JARs
- Third-party JARs

Your baseline must also include SQL scripts to create or update database tables and a subset of `update.xml` to update repository objects that are not included in your baseline.

Each iteration that includes a sample database table upgrade script requires changes to your overall upgrade procedure. You can simply run the database table upgrade scripts in the correct order or concatenate the scripts, but you must modify each sample script appropriately for your environment.

You might find that it is more convenient, more efficient, and ultimately safer to write a single database table upgrade script that is cumulative. In other words, write a single script that combines all of the processing that would have been done by each of the individual database table upgrade scripts if you executed those scripts in the proper order.

Executing a single database upgrade script simplifies the upgrade procedure and gives you the opportunity of eliminating redundant processing, such as creating indexes for one version of Identity Manager and then later dropping and re-creating the same indexes for another version of Identity Manager.

You can also identify an appropriate subset of Identity Manager's `update.xml` in each iteration that is required to update objects in the Identity Manager repository that are not managed as part of the baseline. For a skip-level upgrade, you must ensure this subset of Identity Manager's `update.xml` is cumulative.

NOTE If you are developing a skip-level upgrade, you must be sure to add any configuration objects that were changed by an *updater* to your Identity Manager baseline.

An updater is a program supplied with Identity Manager that updates configuration objects. The updater is invoked by an `ImportCommand` within `update.xml` or within a file that `update.xml` includes. An Updater generally works only with the version of Identity Manager with which it was shipped. Because you are writing a "skip-level" upgrade, the Updater probably will not work with the target version of Identity Manager. Adding any changed configuration object to your baseline is by far the safest approach.

Appendix B

Assessment Worksheets

Having a good understanding of your current Identity Manager installation and the latest Identity Manager version helps you determine an appropriate upgrade method and strategy.

Use the worksheets provided in this appendix to record important configuration data, and then use this data to prepare for and choose an upgrade path.

- Platform Inventory
- Identity Manager Installation
- Custom Components

NOTE If you are using the Identity Manager IDE or an older form of CBE, their automated tools already capture much of this information for you.

Platform Inventory

Record inventory information in Table B-1.

Table B-1 Inventory Information

Platform Component	Your Information
Application server	Version and Service Packs ______________________________ ______________________________ ______________________________ Server Platform Version and Service Packs ______________________________ JDK ______________________________
Database server	Version and Service Packs ______________________________ ______________________________ ______________________________ Server Platform Version and Service Packs ______________________________
Sun Identity Manager Gateway server	Version ______________________________
Java Runtime Environment (JRE)	Version ______________________________
Web Servers	Version and Service Packs ______________________________ ______________________________ ______________________________

Record resource information in Table B-2.

Table B-2 Resource Information

Resource	Name	Version and Service Packs
Resource A	______________________	______________________
		______________________
		______________________
Resource B	______________________	______________________
		______________________
		______________________
Resource C	______________________	______________________
		______________________
		______________________
Resource D	______________________	______________________
		______________________
		______________________
Resource E	______________________	______________________
		______________________
		______________________
Resource F	______________________	______________________
		______________________
		______________________
Resource G	______________________	______________________
		______________________
		______________________
Resource H	______________________	______________________
		______________________
		______________________

Table B-2 Resource Information *(Continued)*

Resource	Name	Version and Service Packs
Resource I	____________________________	___________________________
		____________________________
		____________________________
Resource J	____________________________	____________________________
		____________________________
		____________________________

Identity Manager Installation

Record Identity Manager installation information in the Table B-3.

Table B-3 Identity Manager Installation Information

Identity Manager	Your Information
Installed Version	______________________________
Installed Service Packs	______________________________
	______________________________
	______________________________
	______________________________
	______________________________
Installed Hotfixes	______________________________
	______________________________
	______________________________
	______________________________
	______________________________
	______________________________
	______________________________
	______________________________

Custom Components

Record custom work information in Table B-4. Custom work consists of any component that has been modified for your installation.

Table B-4 Custom Work Information

Component	Your Information
File-System Objects	
Modified JSPs	____
	____
	____
	____
Modified Waveset.properties File	____
	____
	____
	____
Modified wpmessages.properties File	____
	____
	____
	____
Other Customized Property Files	____
	____
	____
	____
Customized Resource Adapters (and Other Custom Java)	____
	____
	____
	____
	____
	____
	____

Table B-4 Custom Work Information *(Continued)*

Component	Your Information
Modified Stylesheets	____ ____ ____ ____
Repository Objects	
Modified Forms	____ ____ ____ ____
Modified Workflows	____ ____
Modified Email Templates	____ ____ ____ ____
Custom Repository Schema	____ ____
Other Custom Repository Objects:	
• Admin groups	____ ____
• Admin roles	____ ____
• Configurations	____ ____

Table B-4 Custom Work Information *(Continued)*

Component	Your Information
• Policies	____________________ ____________________
• Provisioning tasks	____________________ ____________________
• Remedy configurations	____________________ ____________________
• Resource actions	____________________ ____________________
• Resource forms	____________________ ____________________
• Roles	____________________ ____________________
• Rules	____________________ ____________________
• Task definitions	____________________ ____________________
• Task templates	____________________ ____________________
• User forms	____________________ ____________________ ____________________ ____________________

Index

SYMBOLS

$WSHOME/bin directory 33, 39
$WSHOME/patches directory 16, 33, 35, 41
@deprecated tags 5

A

acceptance testing 53, 55
Active Sync processes
  restarting 44
  restoring 51
  starting 46
  stopping 29, 46
adapters, custom 19
analyzing
  deprecation messages 42
  upgrade changes 40, 64
APIs, version changes 19
Application Server, Sun Java™ System 43, 51
application servers
  documenting 15
  rebuilding libraries 42
  restarting 43, 51
  using JDBC data sources 49
application, Identity Manager 7
applications
  configuring 54
  redeploying web 43, 51
archived hotfixes location 33, 35
assessment tools 16

B

backups 35, 47
backward-compatibility 5, 43
baseline
  component customizations 18
  description 9
  resetting Development environment 27
  updating 64
Best practices 14, 29, 40, 44, 46, 48, 50, 51, 62

C

CBE
  description/purpose 8
  documenting your installation 14
  generating an image 9
changedFileList file 41
checklist, upgrade procedure 26
classes
  deprecation policy 4
  hotfix 29, 47
  rebuilding custom 42
  recompiling 5, 19
classes directory 29, 47

commands
- `chmod -R +x *` 33
- `gateway` 15, 35
- `install -nodisplay` 34
- `inventory` 19
- `lh` 15, 16
- post-process 37, 38
- pre-process 37, 38
- to set the environment 36, 38

comparing objects 2
`config` directory 40
Configuration Build Environment. See *CBE.*
configuration description 10
configuration objects
- analyzing changes 40
- baseline 9, 40, 64
- changed by updater program 65
- copying 30
- source control 8
- updating 6
- using SnapShot 30, 39

controlled testing 45
creating simulated resources 45
custom components
- documenting 18
- messages 40
- repository objects 20, 21
- worksheets 72

customizations
- baseline 18
- description 10
- documenting 14
- extracting 40
- file 41, 42
- JSPs 41
- merging 2, 40
- migrating 2
- objects 40, 42
- rebuilding custom Java classes 42
- restoring 41
- restoring after upgrade 42
- tagging and storing 44
- updating 40
- upgrading 5

customized email templates 20

D

data sources, JDBC 49
database servers, documenting 15
database table definitions 18, 50
database table upgrade script 65
database tables
- loading test data 45
- merging changes 44
- replicating for QA environment 54
- updating 32, 40, 46, 64

Debug pages 30
Deprecation policy 4, 42
deprecation warnings 42
Development environment
- description/purpose 7
- resetting 27
- upgrading 28, 34

directories
- `$WSHOME/bin` 33, 39
- `$WSHOME/patches` 16, 33, 35, 41
- `config` 40
- `install` 33
- `logs` 33
- `patches` 16, 33, 35, 41
- `savedObjects` 40, 42

documenting
- current installation information 71
- current platform 14, 68
- custom components 18, 72

downtime, minimizing 57, 59

E

email templates 20
End of Service Life. (*EOSL*) 3
environments
- creating QA 54
- creating Test 45
- descriptions 7

exporting snapshots 31
external processes, stopping and starting 26

F

files, customized 41, 42
`filesNotInstalled` file 41
filesystem objects
 documenting custom 18
 merging changes 44
forms 20
Full Support Phase 4
functional testing 45, 52

G

`gateway` commands 15, 35
Gateway service
 installing 35
 removing 35
 starting 35
 stopping 35
`gateway.zip` file 35
generating an application image 6, 46

H

hotfixes
 archiving 33, 35
 removing 29, 47
 version information 16

I

Identity Manager
 Deprecation policy 4
 End of Service Life (*EOSL*) 3
 Gateway. *See* Gateway service
 installation worksheets 71
 product vs. application 7
 software support policy 3
 upgrading product 32
 verify/document installation components 16
 version 16
Identity Manager application
 generating an image 46
 promoting the image 50
 restarting 52
 stopping 46
 testing 43, 51
Identity Manager IDE plug-ins
 for Eclipse 8
 for NetBeans 8
Identity Manager server 33, 50
Identity Manager servers, restarting 50
`idm.war` files 33
images
 checking in 44
 description 9
 generating 6
 generating with CBE 9
 Identity Manager application 46
 promoting 6, 45, 50
 verifying 50
importing XML 42
`init.xml` files 2
`install` directory 33
`install -nodisplay` command 34
`install.class` files 33
`installed` utility 16
interfaces, deprecation policy 4
inventory
 current installation information 71
 current platform 68
 custom components 72
`inventory` command 19
`inventory` utility 16

J

JAR files, rebuilding 42
Java classes
 deprecation policy 5
 rebuilding custom 42
 recompiling 19

Java Development Kit *See* JDK.
Java methods
  deprecation policy 5
Java Runtime Environment *See* JRE versions.
JDBC data sources 49
JDK
  verifying and documenting versions 15
  verifying platform 45
  versions 68
JDK files
  rebuilding 42
JRE versions 15, 68
JSPs
  merging modified 40
  restoring 41
  updating 19
  using SnapShot 30

L

`lh` commands 15, 16
libraries
  baseline 9
  rebuilding application servers 42
  rebuilding custom classes 42
Limited Support Phase 4
`logs` directory 33

M

manifests 16, 17
manual upgrades 35
merging changes 44
merging customizations 2, 40
messages, customized 40
methods
  deprecation policy 4
  recompiling deprecated 5
minimizing downtime 57, 59
multi-hop upgrade 9, 59

N

nodisplay option 34
`notRestoredFileList` file 41, 42

O

`objectclass` attribute, SPML 21
objects
  renaming customized 40
  restoring customized 42
  `TaskDefinition` 47, 48, 49
operating system
  verifying and documenting versions 15

P

password flags 37
patches
  downloading xiii
  support policy 4
`patches` directory 16, 33, 35, 41
platform
  inventory worksheets 68
  Unix 38
  Windows 36
post-process commands 37, 38
pre-process commands 37, 38
product, Identity Manager 7
Production environment
  description/purpose 7
  special considerations 57
  upgrading 58
promoting images 6, 45, 50
promotion 8
property files 19

Q

QA environment
- description/purpose 7
- replicating database tables 54
- resetting 54
- upgrading 55

R

rebuilding
- custom Java classes 42
- new JAR files 42
- new JDK 42

recompiling
- custom resource adapters 19
- deprecated classes or methods 5

reconciliations
- disabling 29, 46
- re-enabling 44
- restarting 44
- restoring 51

redeploying web applications 43, 51

removing, hotfixes 29, 47

renaming customized objects 40

replicating, Production environment 54

repository objects
- custom 20
- merging changes 44
- renaming 40
- updating objects not in baseline 46

repository schema, custom 20

resetting
- Development environment 27
- QA environment 54
- Test environment 45

resources
- configuration in Test environment 54
- creating simulated 45
- documenting 68
- recompiling custom adapters 19
- requirements for test plan 24

restarting
- Active Sync 44, 51
- application servers 43, 51
- external processes 26
- Identity Manager application 52
- Identity Manager servers 50
- reconciliations 44

restoring customized files 41, 42

S

`savedFiles` directory 41

`savedObjects` directory 40, 42

schema
- Identity Manager version changes 20
- SPML 2.0 considerations 21
- updating 20

`ServerRepository.xml` 49

servers, restarting 50

service packs
- upgrading 2
- version information 16

Services MMC plug-in 35

`set` environment commands 36, 38

simulated resources, creating 45

skip-level upgrade
- description 9
- special considerations 59

Snapshot, using 30, 39

software support policy 3

Solaris
- patches xiii
- support xiii
- verify install directory 33

source-control
- baseline 9, 28, 40, 41, 52
- description/purpose 8
- incorporating fixes 56
- managing SQL Script changes 32
- merging changes 44
- tools 8, 27

special considerations 14, 57, 59

SPML
    objectclass attribute 21
    spml2ObjectClass attribute 21
spml2ObjectClass attribute 21
SQL scripts
    application image 50
    baseline 40, 46, 64
    modifying samples 32
stopping
    external processes 26
    Identity Manager application 46
stylesheets
    documenting 19
    merging modified 40
Sun Identity Manager Gateway
    installing the Gateway service 35
    removing Gateway service 35
    starting the Gateway service 35
    upgrading 34, 35
    verify/document version 15
Sun Java™ System Application Server 43, 51
support, Solaris xiii
supported resources, verifying 15
SystemConfiguration object 40

T

target environments, upgrade procedure for 26
TaskDefinition object 47, 48, 49
TaskInstances 47
Test environment
    creating 45
    description/purpose 7
    resetting 45
    testing after upgrade 52
test plans, developing and executing 24
testing
    controlled 45
    functional 45, 52
    Identity Manager application 43, 51
    Test environment 52
    user acceptance 53
troubleshooting the upgrade 33

U

update.xml files 2, 33, 40, 42, 46, 49, 50, 60, 64, 65
update.xml scripts 10
updater programs 2, 65
updating database tables 32, 40, 46, 64
upgrade log files 33
upgrade paths
    choosing 2, 63
    required 2
upgrade procedure
    description 10
    developing and executing 25
upgrade process description 10
upgrade project 11
upgrade script, database table 65
UpgradePostProcess procedure 38
upgrading
    analyzing changes 40
    choosing a version 22
    Development environment 28, 34
    Identity Manager product 32
    manually 35
    phases 11
    Production environment 58
    QA environment 55
    reasons to upgrade 1
    SPML 2.0 implementations 21
    Sun Identity Manager Gateway 34
    troubleshooting 33
    using Identity Manager upgrade program 34
    using nonstandard techniques 2
user acceptance testing 53, 55

V

verifying images 50
verifying versions
    current Identity Manager installation 16
    hotfixes 16
    JDK 15
    operating system 15
    platform components 14

releases 16
service packs 16
Sun Identity Manager Gateway 15
versions
API changes 19
choosing 22, 63

W

Waveset.properties file 19
web applications, redeploying 43, 51
web servers, verify/document version 15
web.xml 43, 51
workflows 20
worksheets
custom components 72
Identity Manager installation 71
platform inventory 68
WPMessages.properties file 19, 40
WSHOME 49

X

XML
exporting to 31
importing 42
XPRESS changes 43