OUM

DO.070 User Guide

Oracle Consulting Services

Oracle Identity and Access Management

Author: Dieter Steding

Creation Date: May 29, 2018

Last Updated: June 17, 2018

Document Ref: OC\_DE/xxxx/DO.070/001>

Version: 1.0.0.0

**Approvals:**

|  |  |
| --- | --- |
| <Approver 1> |  |
| <Approver 2> |  |

# Document Control

## Change Record

3

| Date | Author | Version | Change Reference |
| --- | --- | --- | --- |
| 29-May-18 | Dieter Steding | 1.0.0.0 | No Previous Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Reviewers

| Name | Position |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# Preface

This document embodies a unified set of standards to be used on IAM projects throughout Oracle Consulting. It addresses the core constructs and structures of the Java language, rather than application of the language (a set of guideline documents are being prepared for the major Java-related technologies within Oracle such as the ADF and IAM frameworks). The standards will be revisited and reissued regularly as they are adopted and exercised by new projects.

This document assumes that the reader has a basic knowledge of Java. Its purpose is not to teach the language but to advise best practice with regard to maintainable and robust code.

## Background

Development of code on any project should be considered a team activity. Any individual developer does not own code, but will more likely evolve through successive iterations of design and build often being updated by several people. As a result it is imperative that developers use guidelines to aid the development process.

These Java coding standards supersede the existing draft Oracle Java coding standards [1] when used within Oracle Consulting. (No work is being done on the draft standards, which will never progress from draft status and will not be updated to reflect the current Java specification).

This document should be used in conjunction with the Sun Java Coding Standards [3]. In some instances it overrides recommendations from Sun, on the basis of past Oracle project experience. When this document fails to cover a topic, recommendations from Sun should be followed.

## Related Documents

1. Draft Oracle Java Coding Standards  
   (http://www-apps.us.oracle.com/java/codestand.htm)
2. How to Write Doc Comments For Javadoc  
   (http://java.sun.com/products/jdk/javadoc/writingdoccomments/index.html)
3. Sun Java Coding Standards  
   (http://java.sun.com/docs/codeconv/index.html)
4. Thinking In Java, Bruce Eckel  
   (http://www.bruceeckel.com/)

### Your Comments are Welcome

<Company Short Name> values and appreciates your comments as a user and reader of this manual. As we write, revise, and evaluate our documentation, your comments are the most valuable input we receive. If you would like to contact us regarding comments and suggestions on future releases of <Subject>, please use the following address or telephone numbers:

<Company Long Name><Contact Address>  
<Contact Phone Number> Email: <Contact Email>

Contents

1 Document Control ii

1.1 Change Record ii

1.2 Reviewers ii

2 Preface 1

2.1 Background 1

2.2 Related Documents 1

3 Repositories and Workspace Structure 4

3.1 Overview 4

3.2 Configuring Subversion Repository 6

3.3 Recommended File System Structure 7

4 Additional JDeveloper Configuration 8

4.1 Add a memory monitor 8

5 Extend Oracle JDeveloper Configuration 9

5.1 Tuning 9

5.2 Deploy Custom Extension 11

5.3 Remote Deployment 14

5.4 Subversion Support 15

5.5 Configuring Proxy Support 16

5.6 Create a new Repository Connection 17

5.7 Default Project Settings 19

5.8 Configuring Features and Preferences 19

6 Java Coding Conventions 29

6.1 General Guidelines 29

6.2 Miscellaneous Guidelines 31

7 Manage Applications 38

7.1 Create an Application 38

7.2 Review the Configuration 45

8 Manage Deployment Projects 46

8.1 Create a Deployment Project 46

8.2 Review the Configuration 53

8.3 Sanity Check 55

9 Exporting Object from Oracle Identity Manager 57

9.1 About the Deployment Directory 58

9.2 Configuring an Export Target 58

10 Appendix A 61

11 Glossary 63

12 Open and Closed Issues 64

12.1 Open Issues 64

12.2 Closed Issues 64

# Repositories and Workspace Structure

## Overview

The project use Global Consulting Delivery Portal (GCDP) for issue tracking and code sharing.

### What is Global Consulting Delivery Portal

The code repository is CollabNet’s Enterprise Edition. This tool:

* Is a hosted, web-based, collaborative project management and development environment
* Will allow project teams to gain greater control of their projects and processes via standardized and consistent delivery that supports our Project Management Standard Operating Procedures (PMSOPs) and Methods
* Provides a distributed development environment, and is available to all including our subcontractors and clients

### Benefits to Projects and Members

GCDP provides the Project Managers (PM's) and team members with a unified platform to collaborate and manage their work more effectively by enhancing:

* Work Plan Management this allows the PM to upload Microsoft Project files and provides the team real time visibility to the project progress, milestones, and their own tasks
* Application Lifecycle Management a customizable Project Home page features quick access to Project artifacts and queries, Project Management and Implementation Method Materials, and Project Library upload and versioning
* Document and Code Management utilizing 'Subversion', an open-source version control mechanism, allows multiple ways to manage documentation and code
* Project Tracking through the use of standardized risks, issues, problems, change requirements, and action workflow enabled templates
* Project Metrics which contains user customizable queries and reports with graphical representation of all artifact's attribute data residing within the OCDP tool
* Communication through the use of Project Announcements, Discussion Forums and Mailing Lists; all communication is archived Addition of an enhanced WIKI capability, new navigation and search functions

To make it easier for all participant of the project you should follow the guidelines for creation of the workspace file system structure.

The framework assumes a specific File System Structure This structure enables to:

* Centralize the common Oracle Consulting artifacts
* Separate the customer project artifacts

This structure allows working on several customer projects where the code repositories are not always on the same server. The framework itself is expecting this workspace structure mentioned in File System Structure for its own purpose. Therefore, you will find in the version control systems the same hierarchy.

This was required to make it possible to handle to different code repositories that the framework has seen during its lifetime. Each of the folders checked in a Subversion Repository. It is not possible to point in a folder hierarchy to different code repositories (or better it is not useful to spread a hierarchy across different Subversion Repositories).

If you do so, it will not be possible to tag or branch the development artifacts.

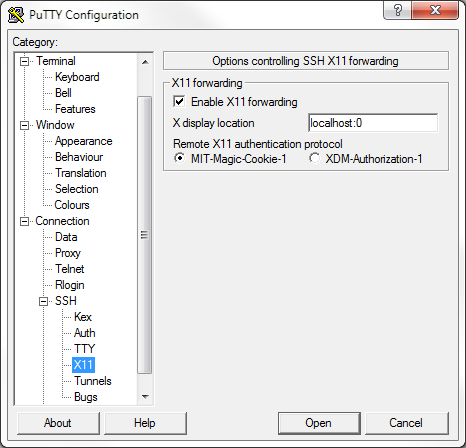
### Prepare your Workstation

It is useful to run the server component of Oracle Identity and Access Management infrastructure components in a Virtual Machine. This avoids the effort to install for each team member a complete environment.

#### Telnet/SSH Client

Most of the environments deployed on UNIX Systems. To manage the Oracle Software Components you will need a SSH Client. The preferred client is **PuTTY** a free implementation of Telnet and SSH for Win32 and UNIX platforms, along with an xterm terminal emulator. [Download](http://the.earth.li/~sgtatham/putty/latest/x86/putty.zip) the ZIP-archive and unpack it in a directory of your choice.

If you want to use the graphical environment for the UNIX systems each time you configuring a new connection don’t forget to enabling X11 forwarding to the host. To do that you navigate in the **PuTTY** Configuration to **Connection** | **SSH** | **X11** and switch on the option **Enable X11 forwarding**.



#### FTP Client

To transfer files to the UNIX systems you will need a SFTP client and FTP client. Legacy SCP protocol should also supported by the client. The preferred client is WinSCP an open source free SFTP client and FTP client for Windows. Legacy SCP protocol is also supported. Its main function is safe copying of files between a local and a remote computer.

[Download](http://winscp.net/download/winscp418.exe) the executable in a directory of your choice.

#### Graphical UNIX Client

There are two options to access the UNIX environments:

* **Virtual Network Computing (VNC)**  
  a graphical desktop sharing system that uses the RFB protocol to remotely control another computer.  
  It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.
* **Xming X Server**  
  provides the basic framework, or primitives, for building such GUI environments: drawing and moving windows on the screen and interacting with a mouse and/or keyboard. X Windows does not mandate the user interface — individual client programs handle this. As such, the visual styling of X-based environments varies greatly; different programs may present radically different interfaces. X windows is built as an additional application layer on top of the operating system kernel.

|  |  |
| --- | --- |
| note_edit | We recommend using the [Xming X Server](http://downloads.sourceforge.net/xming/Xming-6-9-0-31-setup.exe?use_mirror=mesh) free unlimited X Window server for Microsoft Windows (XP/2003/Vista).  It is fully featured, small and fast, simple to use. |

|  |  |
| --- | --- |
| note_edit | We recommend also to install [optional font packages](http://sourceforge.net/projects/xming/files/Xming-fonts/7.5.0.25/Xming-fonts-7-5-0-25-setup.exe/download) for Microsoft Windows (XP/2003/Vista).It is fully featured, small and fast, simple to use.  It solves some problems that you will face using the java console dbca. |

#### Subversion Client

For some activities, you need a separate Subversion Client to manage the Repository. Following Subversion Clients are available.

* TortoiseSVN  
  is a free open-source client for the Subversion version control system. TortoiseSVN integrates seamlessly into Windows Explorer.
* CollabNet Subversion Client  
  is an enterprise-ready distribution of Subversion that includes certified binaries, platform-specific installers.

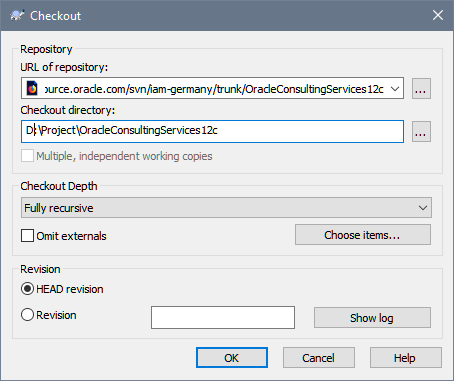
|  |  |
| --- | --- |
| note_edit | We recommend using TortoiseSVN on Microsoft Windows machines.  Please follow Setup TortoiseSVN to install and configure initially. |

## Configuring Subversion Repository

The code repository of Oracle Consulting Framework designed for the Oracle Identity and Access Management Product Family10g is still located in the Sample Code Repository of Oracle Technology Network. The code repository of Oracle Consulting Framework designed for the Oracle Identity and Access Management Product Family11g/12c product line goes to [Global Consulting Delivery Portal](https://gcdp.oracle.com/).

The Workspace provides the source and binaries and can checked out from the repositories mentioned above.

To checkout the tree, perform the following steps:

1. From the Windows Explorer navigate to D:\Project for example
2. With a right-mouse-click, open the context menu for this folder.
3. Choose **SVN Checkout...** to open the Checkout Dialog.
4. As **URL of repository** type in  
     
   <https://gcdpsource.oracle.com/svn/iam-germany/trunk/OracleConsultingServices12c>  
     
   

## Recommended File System Structure

To make it easier for all participant of the project you should follow the guidelines for creation of the Workspace Structure. The Workspace Structure consists of several directories:

* **FoundationFramework**  
  the artifacts shared across products
* **IdentityManager**  
  the artifacts related to Oracle Identity Manager
* **AccessManager**  
  the artifacts related to Oracle Access Manager
* **IdentityAnalytics**  
  the artifacts related to Oracle Identity Analytics
* **InternetDirectory**  
  the artifacts related to Oracle Internet Directory
* **UnifiedDirectory**  
  the artifacts related to Oracle Unified Directory
* **VirtualDirectory**  
  the artifacts related to Oracle Virtual Directory

# Additional JDeveloper Configuration

## Patching

Patch 27410841: JDEV PERFORMANCE ISSUE WHEN ADF CODE AUDIT CALLS ORACLE.BALI.XML.MODEL.\*

Patch 28151020: ADF BUNDLE PATCH 12.2.1.3.0(ID:180607.1214.S)

## Add a memory monitor

# Extend Oracle JDeveloper Configuration

The purpose of the guide is to provide comprehensive team member Orientation Guide to use during Project Kickoff and afterward for the on boarding of new team members. This will help to streamline the on-boarding process, ensure consistent communications and reduce the project manager and team leads workload during the on-boarding process.

## Tuning

If you feel your JDeveloper 12c is too slow, follow these 3 basic steps and increase performance of JDeveloper 12c.

### Configure Jdeveloper memory settings in ide.conf

Open in a text editor the file <JDEVELOPER\_HOME>/jdeveloper/jdev/bin/jdev.conf and set

# optimize the JVM for strings / text editing

AddVMOption -XX:+UseStringCache

AddVMOption -XX:+OptimizeStringConcat

AddVMOption -XX:+UseCompressedStrings

# if on a 64-bit system, but using less than 32 GB RAM, this reduces object pointer memory size

AddVMOption -XX:+UseCompressedOops

# use an aggressive garbage collector (constant small collections)

AddVMOption -XX:+AggressiveOpts

# for multi-core machines, use multiple threads to create objects and reduce pause times

AddVMOption -XX:+UseConcMarkSweepGC

AddVMOption -DVFS\_ENABLE=true

AddVMOption -Dsun.java2d.ddoffscreen=false

AddVMOption -XX:+UseParNewGC

AddVMOption -XX:+CMSIncrementalMode

AddVMOption -XX:+CMSIncrementalPacing

AddVMOption -XX:CMSIncrementalDutyCycleMin=0

AddVMOption -XX:CMSIncrementalDutyCycle=10

### Configure JVM settings in jdev.conf

Open in a text editor the file <JDEVELOPER\_HOME>/jdeveloper/ide/bin/ide.conf and set

# Set the default memory options for the Java VM which apply to both 32 and 64-bit VM's.

# These values can be overridden in the user .conf file, see the comment at the top of this file.

AddVMOption -Xms2048M

AddVMOption -Xmx4096M

# Load the core extension at startup

AddVMOption -Doracle.ide.startup.features=oracle.ocs.workspace.iam

# optimize the JVM for strings / text editing

#AddVMOption -XX:+UseStringCache

AddVMOption -XX:+OptimizeStringConcat

AddVMOption -XX:+UseCompressedStrings

# if on a 64-bit system, but using less than 32 GB RAM, this reduces object pointer memory size

AddVMOption -XX:+UseCompressedOops

# use an aggressive garbage collector (constant small collections)

AddVMOption -XX:+AggressiveOpts

# for multi-core machines, use multiple threads to create objects and reduce pause times

AddVMOption -XX:+UseConcMarkSweepGC

AddVMOption -DVFS\_ENABLE=true

AddVMOption -Dsun.java2d.ddoffscreen=false

AddVMOption -XX:+UseParNewGC

AddVMOption -XX:+CMSIncrementalMode

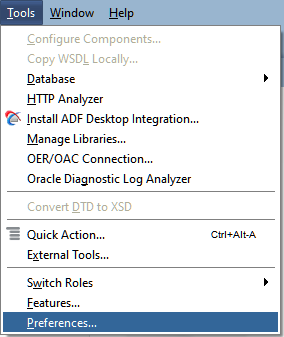
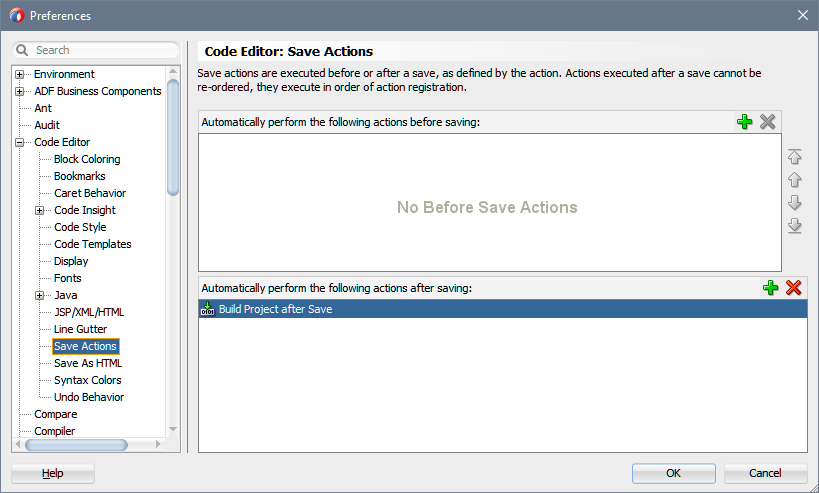
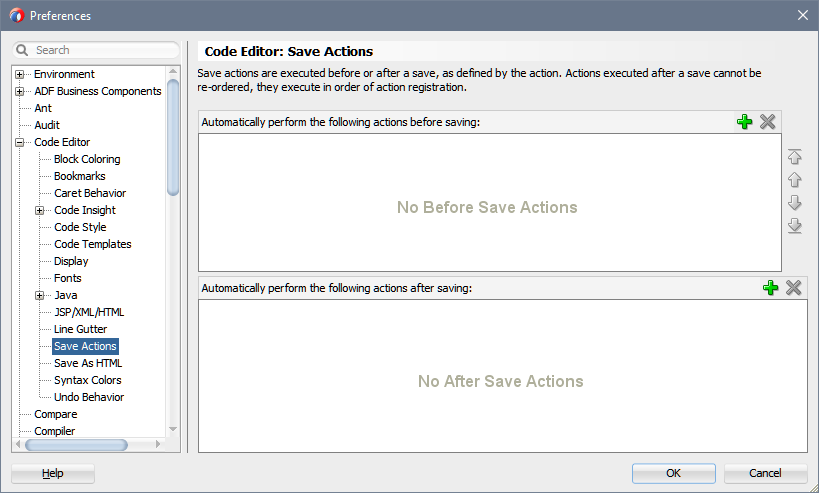
AddVMOption -XX:+CMSIncrementalPacing

AddVMOption -XX:CMSIncrementalDutyCycleMin=0

AddVMOption -XX:CMSIncrementalDutyCycle=10

### Reduce ON-Save and AFTER-Save Actions

To reduce ON Save and AFTER Save Actions:

1. Open Oracle JDeveloper 12c
2. Navigate to the menu **Tools** | **Preferences…**  
     
     
     
   This opens the Preferences Dialog in Oracle JDeveloper 12c.
3. In the **Category** navigation tree drill down to the option **Code Editor**. Open this node and select **Save Actions**. You should see some information like shown in the following figure:  
     
   
4. From the display **Automatically perform the following actions after saving:** remove the entry **Build Project after Save**.  
     
   

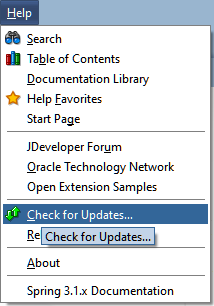
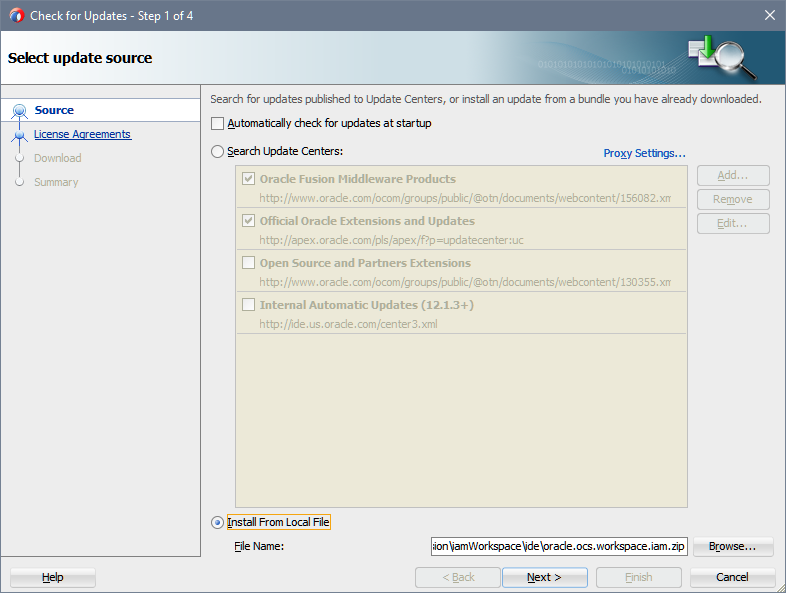
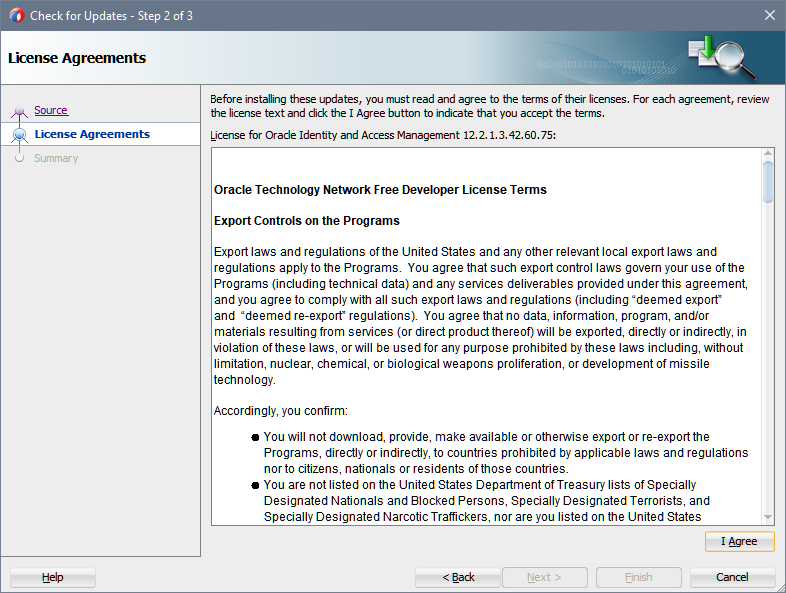
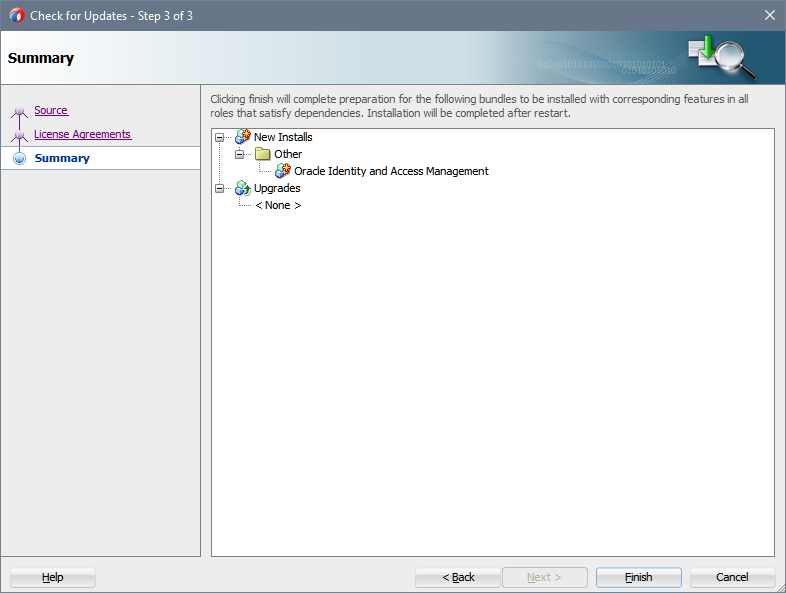
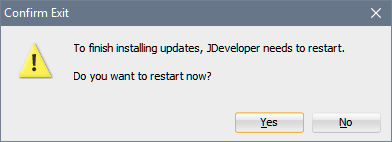
## Deploy Custom Extension

This section describes the installation and usage of the Oracle Consulting JDeveloper plug-in Extension.

You can install the extension directly from inside JDeveloper through the Help | Check for Updates menu option which is the recommended way to install extensions.

The extension itself not provided by the official Oracle JDeveloper Extension Exchange portal. You cannot connect to the internet from your JDeveloper, to download the extension from this page, and point the Check for Updates wizard to the local file you have downloaded. Use this link instead to download the extension bundle.

To make the required extension available of Oracle JDeveloper 12c perform following steps:

1. Open Oracle JDeveloper 12c
2. Navigate to the menu **Help** | **Check for Updates…**  
     
     
     
   This opens the Update Wizard of Oracle JDeveloper 12c.
3. In the Select update source step choose the option **Install From Local File** and specify the location of the package in the field **File Name**  
     
     
     
   Press **Next**
4. In the License Agreements review the license and click **I Agree** to indicate that you accept the terms.  
     
     
     
   Click **Next**
5. After the archive analyzed by the Update Wizard review in the Summary Screen that you picked up the correct version  
     
   y  
     
   Press **Finish** to close the Update Wizard.  
   You will be ask for to restart Oracle JDeveloper 12c.  
     
     
     
   Press **Yes** to restart.

## Remote Deployment

Most of the build tools used to prepare and deliver the deployments to a component relaying on ANT-Tasks.

One of those tasks copies a file or set of files to a (remote) machine running an SSH daemon. This task depends on external libraries not included in the Ant distribution.

If you want to use those tasks to ease, your live download the latest Java Archive from [SourceForge](http://sourceforge.net/projects/jsch/files/) Repository. Alternatively, you can also use the jsch-0.1.53.jar from <FMW\_ORACLE\_HOME>/oracle\_common/modules/thirdparty.

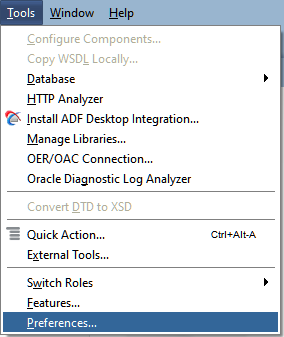
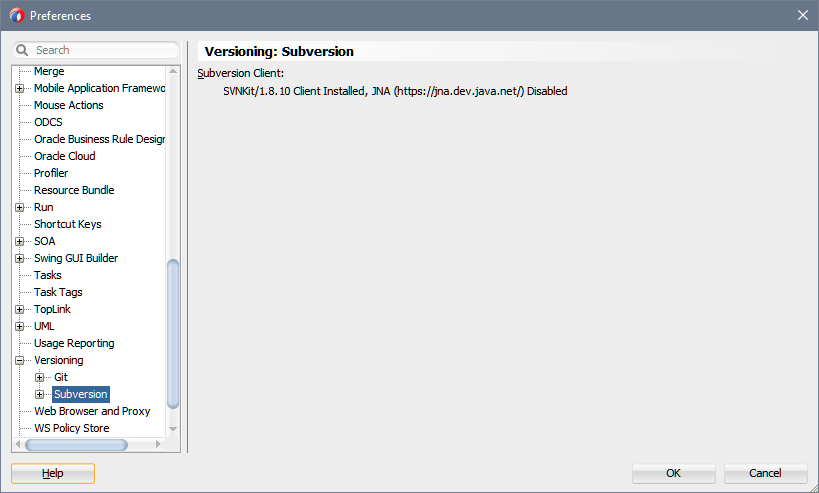
Copy the JAR-file to

<FMW\_ORACLE\_HOME>/oracle\_common/modules/thirdparty/org.apache.ant/1.9.8.0.0\apache-ant-1.9.8/lib

Restart JDeveloper after you have copied the file to the specified location.

## Subversion Support

Validate Version Support

1. Open Oracle JDeveloper 12c
2. Navigate to the menu **Tools** | **Preferences…**  
     
     
     
   This opens the Preferences Dialog in Oracle JDeveloper 12c.
3. In the **Category** navigation tree drill down to the option **Versioning** to check the current version of the extension. At the time being this extension might not be loaded, hence press **Load Extension** to make the extension available. You should see some information like shown in the following figure:  
     
   

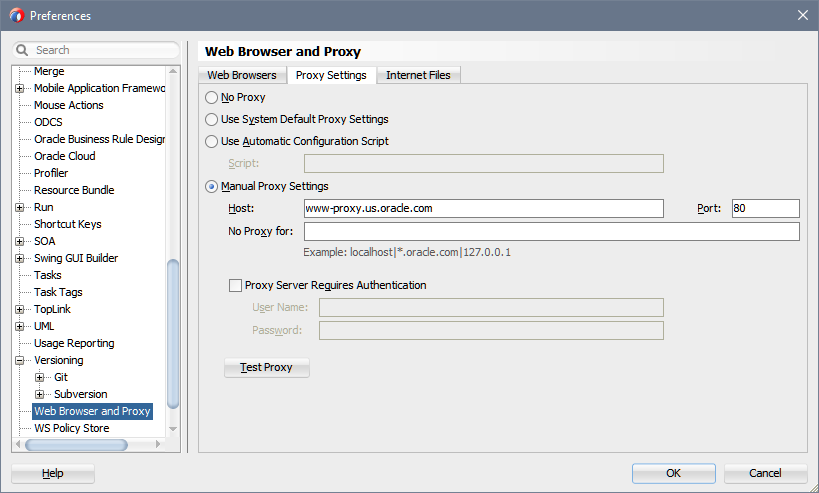
## Configuring Proxy Support

If you work inside of the Oracle Intranet and in some customer networks you will need to connect to Global Consulting Delivery Portal through a Forward Proxy.

This proxy has to be configured in **Tools** | **Preferences** | **Web Browser and Proxy**. In the dialog that appears switch to **Proxy Settings** and provide following information:

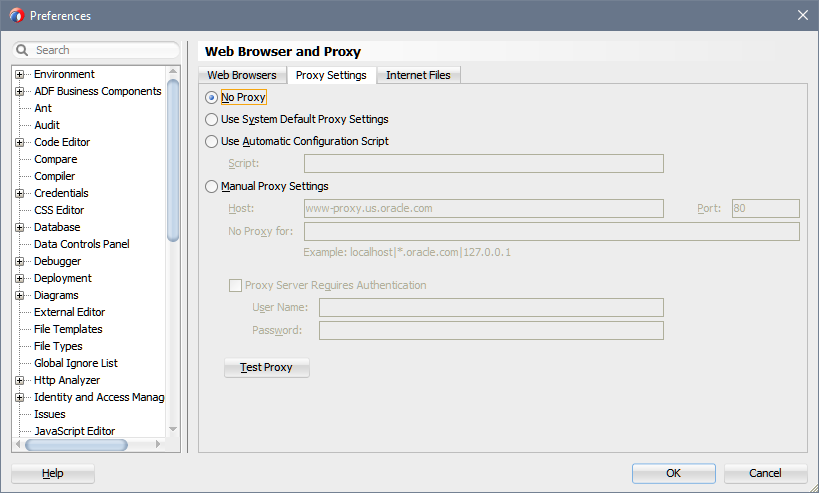
* From inside the Oracle network (including VPN)

|  |  |
| --- | --- |
| **Manual Proxy Settings** | check |
| **Host Name** | www-proxy.us.oracle.com |
| **Port Number** | 80 |
| **Exceptions** | <clear out> |



* From outside the Oracle network (including VPN)

|  |  |
| --- | --- |
| **No Proxy** | check |



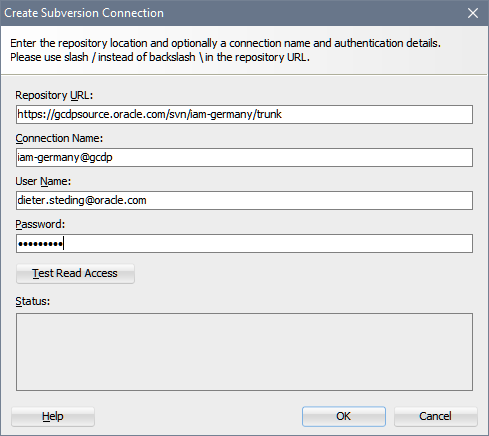
## Create a new Repository Connection

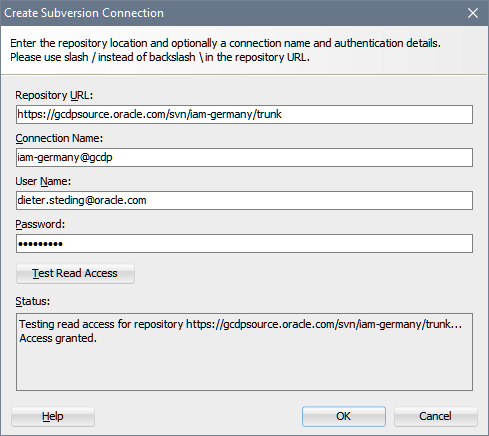
To use the versioning features of Oracle JDeveloper 12c you need to create at first a new Repository Connection.

Perform the following steps to create a new Repository Connection for any new project:

1. Open the Subversion Navigator by **Team** | **Subversion** | **Create Subversion Connection…**.  
   In the dialog that follow up select Manually **Create a Subversion Connection**.
2. In the dialog that appears provide following information:

|  |  |
| --- | --- |
| **Repository URL** | https://gcdpsource.oracle.com/svn/iam-germany/trunk |
| **Connection Name** | iam-germany@gcdp |
| **User Name** | Your Single Sign On User Name |
| **Password** | Your Single Sign On Password |



1. Press the button **Test Read Access**
2. The message in the Status text box has to be:  
     
   Testing read access for repository https://gcdpsource.oracle.com/svn/iam-germany/tunk...  
   Access granted.  
     
   
3. Click OK to close the dialogue.

## Default Project Settings

* Project Source Paths

|  |  |
| --- | --- |
| **Java Source Directory** | src |
| **Output Directory** | bin |

* ADFm Sources

|  |  |
| --- | --- |
| **ADFm Source Directory** | adf |

* Modelers

|  |  |
| --- | --- |
| **Modelers** | uml |

* Web Application

|  |  |
| --- | --- |
| **HTML Root Directory** | static |

* Java Doc

|  |  |
| --- | --- |
| **Destination** | doc |
| **Document Tags** | @version @author @since @deprecated |

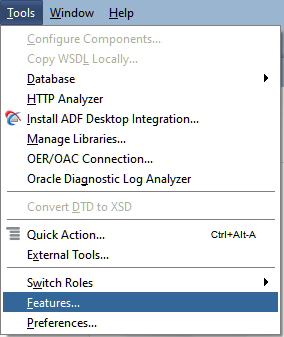
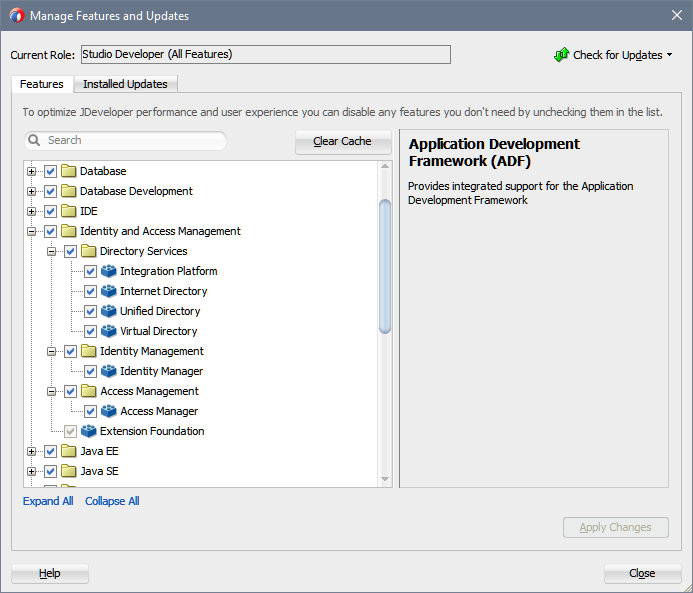
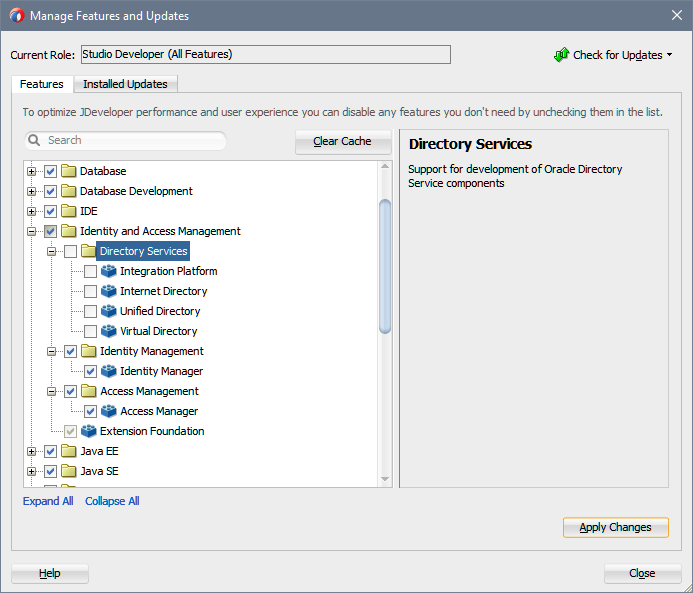
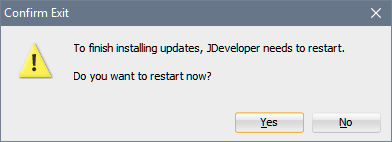
## Configuring Features and Preferences

The extension provides the capabilities to switch off or on specific features.

Most of the features provided by the extension hidden until you have configured the preferences.

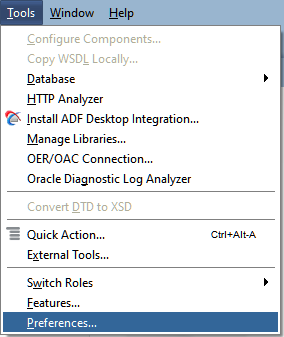
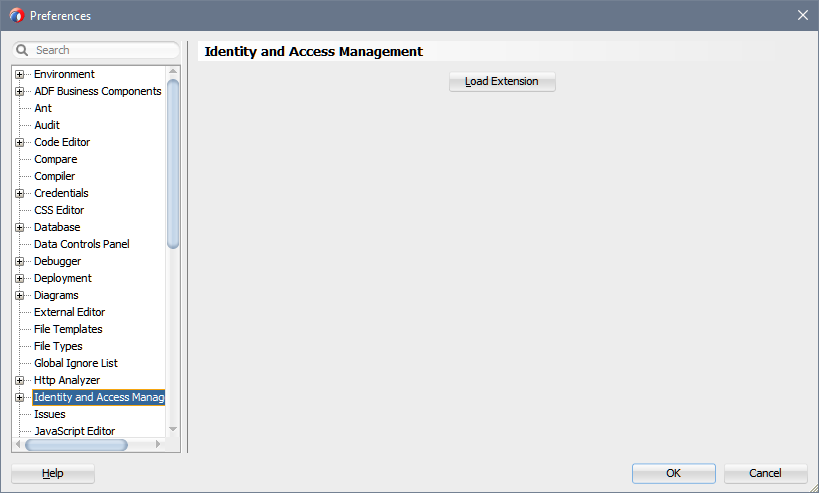
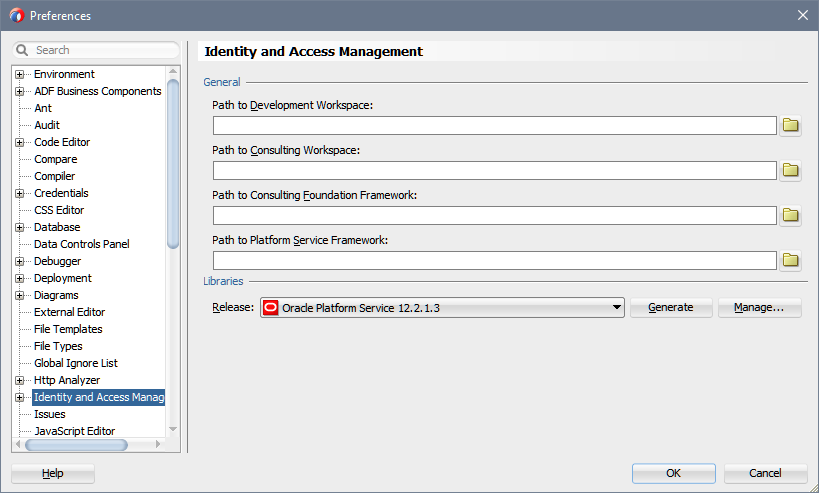
### Configuring the Features

Complete the following steps to configure Oracle Identity and Access Management extension plug-in:

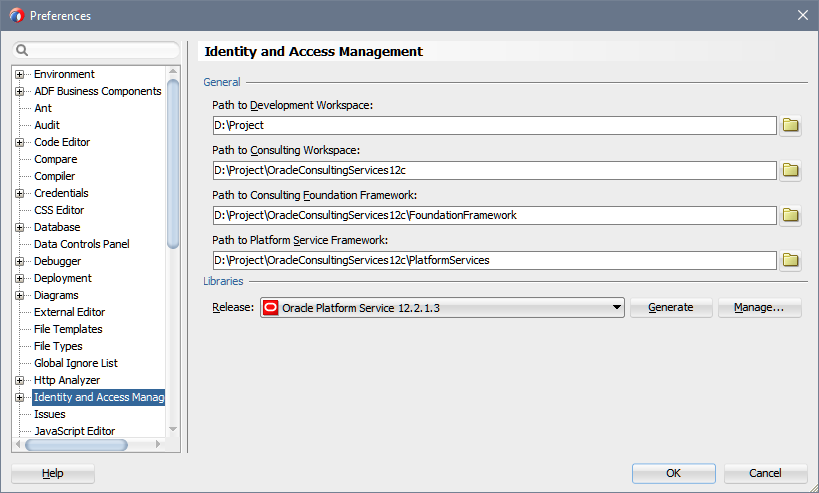
1. Open Oracle JDeveloper 12c
2. Navigate to the menu **Tools** | **Features…**This opens the Manager Features and Update Dialog of Oracle JDeveloper 12c.
3. In the navigation tree scroll down until the feature Oracle Identity and Access Management folder becomes visible and expand the node and sub-nodes.  
     
   
4. Select only the features that you want to use.  
     
     
     
   Press **Apply Changes** to close the Manager Features and Update Dialog.  
   You will be ask for to restart Oracle JDeveloper 12c.  
     
      
     
   Press **Yes** to restart.

### Configuring the Oracle Identity and Access Management Preferences

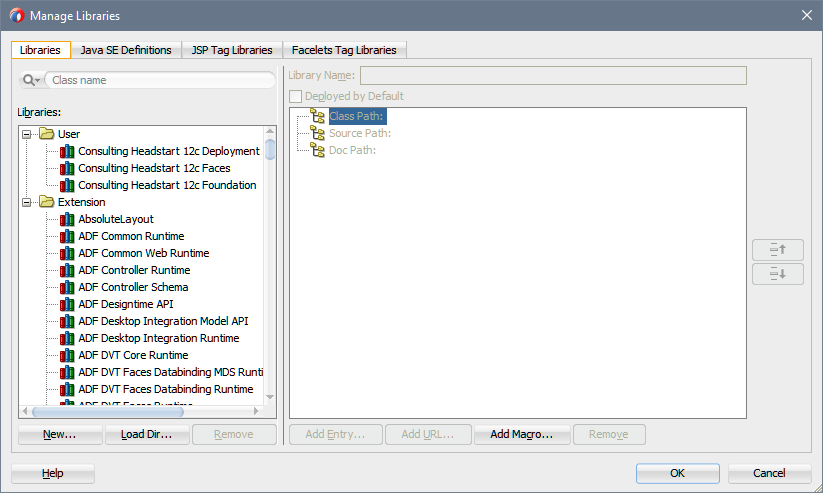
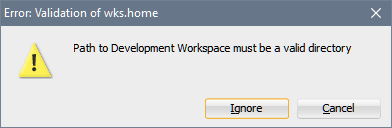
Complete the following steps to configure Oracle Identity and Access Management extension plug-in:

1. Open Oracle JDeveloper 12c
2. Navigate to the menu **Tools** | **Preferences…**  
     
     
     
   This opens the Preferences Dialog in Oracle JDeveloper 12c.
3. In the left hand pane scroll down to the **Oracle Identity and Access Management** preferences and select this node.  
     
     
     
   These displays the preferences that needs to configure and are common for all dependencies. At the time being this extension might not be loaded, hence press **Load Extension** to make the extension available.  
     
   
4. To configure Oracle Identity and Access Management preferences you have to provide in the section General:

|  |  |
| --- | --- |
| **Path to Development Workspace** | The directory folder of your local file system where you will develop.  Either enter the full-qualified path in the text field or use **Browse...** to pick up an existing folder from your local file system.  In the example below, we use the root path as for the development of the Oracle Consulting Services frameworks. |
| **Path to Consulting Workspace** | The directory folders of your local file system where you checked out the workspaces provided by Oracle Consulting Services from Global Consulting Delivery Portal.  If you do not have configured the Subversion Repository, follow the instruction of Repositories and Workspace Structure.  In the field **Path to Consulting Workspace,** you specify the root path of the deliverables. |
| **Path to Consulting Foundation Framework** | If you specify the correct path in **Path to Consulting Workspace,** the field **Path to Consulting Foundation Framework** already populated for you. |
| **Path to Platform Services Framework** | If you specify the correct path in **Path to Consulting Workspace,** the field **Path to Platform Services Framework** already populated for you. |



The options Path to Development Workspace and Path to Consulting Workspace need not to be in the same hierarchy. This allows you to separate customer specific workspaces completely from the Oracle Consulting artifacts.

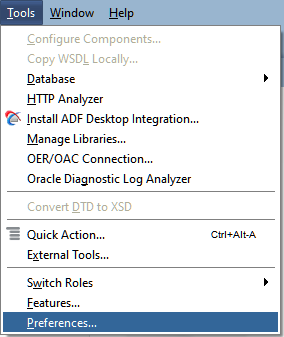
1. After you entered all necessary paths, you should press **Generate** in the section Libraries. By doing so, the libraries that you may assign to your projects created for you. This activity performs also the validation of the paths that you provided in the section **General**. If an error will reported, you need to correct the issue and repeat the generation of the libraries.
2. Validate the existence of libraries created by the step above by clicking **Manage....** This opens the Manage Library dialog. Here you can watch if the libraries are matching your environment. At this time, only the user library  
     
   Consulting Headstart 12c Deployment  
   Consulting Headstart 12c Faces  
   Consulting Headstart 12c Foundation  
     
   are created.  
     
     
     
   If something went wrong close the dialog Manage Library and repeat step 4 until 6.
3. After you, configured **Oracle Identity and Access Management** preferences expand this node in the preference navigator. Now the product specific preference nodes can configured. It is not important in which sequence you will configure these preferences. It is also not important to configure all of them. You will only miss some option in the gallery if a specific product is not configured yet.  
     
     
     
   If this happens correct the path entry mentioned in the Message Box. As long as such errors are not corrected you will not be able to navigate out of this page. The only option that you have is to click on **Cancel** and exit the dialog.

### Configuring the Oracle Identity Manager Preferences

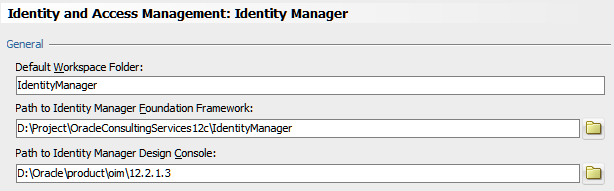
To use the features of Oracle Identity Manager provided by the extension you need to configure the preferences.

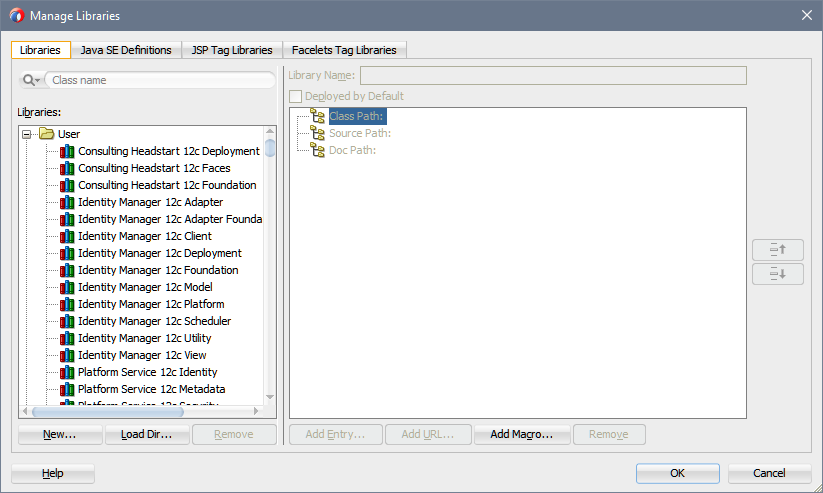
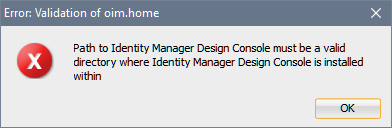
Unless you have it not done now all gallery items hidden.

Complete the following steps to configure Oracle Identity and Access Management extension plug-in:

1. Open Oracle JDeveloper 12c
2. Navigate to the menu **Tools** | **Preferences…**  
     
     
     
   This opens the Preferences Dialog in Oracle JDeveloper 12c.
3. In the left hand pane scroll down and open the **Oracle Identity and Access Management** preferences and expand this node. Select the preference node **Identity Manager** that now is visible.  
     
     
     
   This displays the preferences you need to configure especially for Oracle Identity Manager Features of the extension plug-in. At the time being this extension might not be loaded, hence press **Load Extension** to make the extension properties available.  
     
   Most of the fields pre-populated with default values that derived from the Oracle Identity and Access Management preferences.
4. To configure Oracle Identity Manager preferences you have to provide in the section **Genera**l:

|  |  |
| --- | --- |
| Default Workspace Folder | The directory folder of your local file system where the applications and all projects related to Oracle Identity Manager are located. |
| Path to Identity Manager Foundation Framework | The directory folders of your local file system where you checked out the workspaces provided by Oracle Consulting Services for development of Oracle Identity Manager artifacts.  If you configured **Path to Consulting Workspace** in the preference panel of **Oracle Identity and Access Management** correctly, this path already populated for you. |
| Path to Identity Manager Design Console | The directory folder of your local file system where the Design Console of Oracle Identity Manager is installed.  You have copied the configured installation from your **Oracle Identity and Access Management** installation.  Either enter the full-qualified path in the text filed or use **Browse...** to pick up the folder from your local file system.  The folder that you have to specify is the path to xlclient.cmd without the filename.  The extension plug-in itself does not make use of the command lines but needs the location to validate the installation before it will accept the path. |
| Path to Platform Services Framework |  |



1. In the section **Libraries,** you should choose the correct version of your Oracle Identity Manager installation. From the drop-down **Release** select the appropriate version.  
     
     
     
   After you entered all necessary paths, you should press **Generate** in the section **Libraries**. By doing so, the libraries that you may assign later to your projects related to **Oracle Identity Manager** created. This activity performs also the validation of the paths that you provided in the section **General**. If an error is reported to you correct the issue and repeat the generation of the libraries
2. Validate the existence of libraries created by the step above by clicking **Manage...**.This opens the Manage Library dialog. There you can watch if the libraries are matching your environment.  
     
   
3. After you configured **Identity Manager** Preferences, you can select a different preference node. It is also not important to configure all of them. You will only miss some option in the gallery if a specific product is not configured yet. Leaving the preference node **Identity Manager** validates that configuration. If one of the paths is not correct you noticed by a Message Box like the one below  
     
     
     
   If this happens correct the path entry mentioned in the Message Box. As long as such errors not solved, you will not be able to navigate out of this page. The only option that you have is to click on Cancel and exit the dialog.

# Java Coding Conventions

## General Guidelines

### Formatting code

Use 2 spaces of indentation per nesting level. Use spaces rather than tabs to avoid problems with different tab definitions on printers and editors. The length of each source line should not exceed 80 characters.

JDeveloper should be set up to use these defaults. Go to **Tools** | **Preferences** | **Code Editor** | **Code Style**:

|  |  |
| --- | --- |
| **Indention Size** | 2 |
| **Use Tab Character** | Unchecked |
| **Tab Size** | 2 |

### Comments

Comments must be produced in parallel with source code development, not tagged on afterwards. Comments should be used as follows:

* Javadoc comments (/\*\* \*/) should be used to generate appropriate class, method and variable documentation (see section X for more detail).
* Single-line comments (//) should be used for any non-documentation comments within code. If there are several consecutive lines of comment, each must begin with //.
* C-style comment markers (/\* \*/) should only be used to comment out blocks of code. This should only be done as part of the development process or debugging, delivered modules should not contain any commented out code.

### Source Files

A Java source file may contain only one public class or interface. When top-level (i.e. package level) non-public classes and interfaces are associated with a public class, you can put them in the same source file as the public class. The public class should be the first class or interface in the file.

In addition to top-level classes, defined in the context of the package, inner classes can be defined within the context of a parent class. This includes member classes, local classes and anonymous classes (see [4] for more information). There are some situations in which inner classes are useful (for example, local and anonymous classes are often used in event-handling code). However they can also make the code difficult to follow. In general, consider the justification for an inner class carefully (for example, the class will not be available outside the context of the parent class) and ensure that the inner class is fully commented.

Organize the material in each source file as follows:

1. File header.
2. Package statement.
3. Import statements.
4. Class/interface javadoc comments.
5. Class/interface declaration.
   * Member variable definitions.
   * Constructors.
   * Method definitions.

The final three categories form a repeating group. Public member variables, constructors and methods appear first. This is then followed by protected member variables, constructors and methods, and so on for default (package level) and private visibilities. Within each set of method definitions, group ‘get’ and ‘set’ functions for a particular attribute together and group other logically related methods together.

#### File Header

Begin each file with a non-javadoc comment including:

1. Disclaimer, if applicable
2. Copyright information, if applicable.
3. Information identifying the file including the project, subsystem and filename.
4. A list of the classes or interfaces defined in the file, which will always include the one public class or interface included in the file plus any non-public top-level classes and any inner classes.
5. History table listing dates, authors, and summaries of changes. Change summaries can often be generated automatically by source control systems, for example PVCS replaces the label $Log$ with the change history (although this is only useful if developers enter meaningful comments when checking files in).

For example:

/\*

Oracle Deutschland GmbH

This software is the confidential and proprietary information of

Oracle Corporation. ("Confidential Information"). You shall not

disclose such Confidential Information and shall use it only in

accordance with the terms of the license agreement you entered

into with Oracle.

ORACLE MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE

SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE

IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR

PURPOSE, OR NON-INFRINGEMENT. ORACLE SHALL NOT BE LIABLE FOR ANY DAMAGES

SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING

THIS SOFTWARE OR ITS DERIVATIVES.

Copyright © 2014. All Rights reserved

-----------------------------------------------------------------------

System : Oracle Access Authentication Plug-In Library

Subsystem : Common Shared Runtime Facilities

File : AccessPluginError.java

Purpose : This file implements the interface

AccessPluginError.

Revisions Date Editor Comment

------------+-----------+-----------+-----------------------------------

3.1.0.0 2014-28-10 DSteding First release version

\*/

#### Import Statements

Although Java allows the use of an asterisk as a wildcard character in import statements its use should be avoided, since it increases the chance of namespace clashes between packages. Instead, explicitly import each class required. This also makes maintenance easier, since a fully qualified import statement allows you to find documentation relating to the class quickly.

To improve clarity, imports should be listed in order of increasing package specialization with respect to the project. ‘Java’ imports should be listed first, followed by ‘javax’ imports and any other non-Oracle imports. Then Oracle imports should be listed, again in order of increasing specialization - packages developed on other projects first followed by packages developed on the current project. Multiple imports from a particular package should be listed alphabetically. For example,

import java.util.Stack;

import java.util.Enumeration;

import oracle.abc.Utility;

import oracle.iam.access.Employee;

import oracle.iam.access.Location;

JDeveloper can automatically sort the import statements in a class, removing duplicates and unused imports. Right click in the code editor and select Organize imports… and then one of the options.

Note that in general import statements should be used rather than the use of fully qualified class names within code. This is unwieldy and, since the class name may be mentioned in several places, difficult to maintain as the code evolves. The use of two identically named types (for example, java.util.Date and java.sql.Date) is an exceptional case in which the more frequently used type should be imported and the least used type fully qualified within the code.

## Miscellaneous Guidelines

### Magic Numbers

There should be no hard coded ‘magic numbers’ within source code. All constant values should be declared as static final class variables, and a comment included with the variable definition as to what it is used for.

### Initialization

All variables should be explicitly initialized, at their point of declaration. Without explicit initialization the default type initialization is used, which may cause problems if someone subsequently changes the variable’s type.

One Statement Per Line

Code should be laid such that there is only one statement per line. This improves readability and maintainability and makes debugging using a source code debugger easier. For example the following code fragment:

if (state) { setEnabled(state); }

should be written as follows:

if (state) {

setEnabled(state);

}

### Ternary Operator

Used wisely, the ternary operator yields concise, readable code. It should only be used when the expressions involved are simple and their purpose clear. Parentheses should be used to aid readability. For example,

String stateLabel = (a.isModified())? "Modified" : "";

### ‘Dangling Else’ Problem

Braces must always be used to delimit the branches of an if-else statement, even if the branch consists of a single statement. This avoids the so-called ‘dangling else’ problem where even though the code layout and indentation suggests one mode of execution the reality is different. For example, consider the following code fragment:

if (a == 1)

if (b == 3)

myString = “hello”;

else

myString = “goodbye”;

The indentation suggests that the else clause is intended to match the first ‘if’ (i.e. a == 1) but in fact it will be interpreted in the context of the second ‘if’ (i.e. b == 3). This ambiguity is avoided if braces are used:

if (a == 1) {

if (b == 3) {

myString = “hello”;

}

}

else {

myString = “goodbye”;

}

The same principle applies to other flow control and looping constructs such as switch statements (see section x.x), for, do-while and try-catch statements, in that all blocks of code used in these constructs must be delimited by braces - even if the block only consists of one line of code.

### Switch Statements

The ‘case’ labels in a switch statement should be indented to the next level as the ‘switch’ label- these are all parts of the same statement, much as the ‘if’ and ‘else’ labels in an if-else statement have the same level of indentation.

Each case statement must have its own ‘break’ statement, unless it is explicitly intended that execution fall through to the next case statement. If fall through is intended a comment must be included stating this.

All switch statements should have a default case statement, even if at the time of writing no circumstance is envisaged where the default will be called. As programs evolve domains inevitably change and the presence of a default statement will help catch potential problems.

The code associated with each case statement should be delimited by its own set of braces. This avoids unexpected problems with regard to the scope of variables. For example the following code fragment will not compile, since variable ‘i’ is declared twice at switch-level scope:

...

case 1: int i = 0;

// Some code using i...

break;

case 2: int i = 0; // Will not compile

// Some further code using i...

break;

whereas with the addition of case-level braces the code will compile:

...

case 1: {

int i = 0;

// Some code using i...

}

break;

case 2: {

int i = 0;

// Some further code using i...

}

break;

### Empty Statements

There are situations where a statement is not required even though in general a statement would exist. Typically this involves a looping construct where the necessary processing is done in the predicate of the construct itself. Such scenarios must be fully commented since they may not be immediately obvious to anyone maintaining the code. Also, consideration must be given to rewriting the code in such a way as to improve its legibility. For example

for (int i=0; i<10; a[i++] = i) {

// intentionally left blank

}

could also be written

for (int i=0; i<10; i++) {

a[i] = i + 1;

}

### String Manipulation

Intensive manipulation of strings should be performed using the StringBuilder class rather than String as this is usually more efficient. Instances of String are immutable, and operations such as concatenation are achieved using StringBuilder objects behind the scenes. For example,

String s = new String();

s = "One,";

s = s + "two,";

s = s + "three";

System.out.println(s);

will probably result in the creation of several temporary StringBuilder objects, whilst a single StringBuilder object can be used to achieve the same effect:

StringBuilder builder = new StringBuilder();

builder.append("One,");

builder.append("two,");

builder.append("three");

System.out.println(builder.toString());

### Garbage Collection

In general Java’s garbage collector can be relied upon to reclaim the memory holding an object when no handles to that object remain. However sometimes handles to an object continue to exist even when the object has ceased to be useful (for example, an object created and used at the start of a method but ignored for the rest of the method). When large objects or arrays cease to become useful it is good practice to explicitly set all handles to them to null. This will allow the garbage collector to reclaim the associated memory as soon as possible.

A class can define a finalize() method which will be called by the garbage collector when an instance of the class is being reclaimed. However, Java makes no guarantees about when garbage collection will occur or what order objects will be collected in. Indeed, an object may not be garbage collected at all. This means that the finalize() method is of very little use and in particular should not be used to free non-memory resources such as file descriptors associated with an object.

The try/catch/finally method discussed in section 5.11.1 should be used to free resources instead. For example a database connection should be closed in a finally block so it will be released irrespective of whether the JDBC code completed successfully or not.

### Static Declarations

When the value of a static class variable is changed, all instances of the class (within a single Java Virtual Machine (JVM)) see the new value. Care should be taken when using static member variables to ensure that this is the desired behavior. Note that static final variables are set at the point of declaration (to act as immutable constants) and cannot be changed later.

The use of static methods should also be considered carefully, since static methods can only reference static class variables and for the reasons already discussed these are potentially problematic.

Classes used in a completely static manner are reasonable when implementing pools of objects, shared resources and factory methods. However bear in mind that a static attribute is only static within a single JVM, which means they cannot be relied upon in multi-server and multi-process environments such as Oracle Application Server. To ensure that classes such as these are only used in a static manner and never instantiated, the class constructor can be declared private.

When referring to a static attribute or class method, use the form

ClassName.ATTRIBUTE\_NAME

rather than using an object reference. This is more efficient and emphasizes the static nature of the call.

### Logging

All classes should have sufficient logging. A class without logging is impossible to debug in a production environment. Some of the points that should be logged are:

* Class creation.
* Lifecycle methods e.g. ejbCreate and ejbRemove.
* All exceptions should be logged.
* Method return values.

Modern logging frameworks allow the level of logging to be configured at runtime so overhead on performance is limited. It is recommended that the Apache Commons Logging API be used as it abstracts the actual logging implementation used. It will automatically use Log4J, J2SE 1.4 Logging or stdout and allows the developer to add logging statements to code during development and the application assembler to specify a logging sub system later.

Each class should have a static reference to an org.apache.commons.logging.Log object:

private static Log log = LogFactory.getLog(ServiceLocator.class);

Each logging statement should be of the form:

if (log.isDebugEnabled()) {

log.debug(“Message:” + this.property);

}

The guard statement reduces the overhead of the statement by only carrying out the String concatenation if the log is enabled at the correct level.

### Exception Handling

Make sure that exceptions are handled at the correct level. If a method does not have sufficient knowledge to know how to treat an exception, it should pass it up to its calling method to deal with.

Catch blocks should only handle checked Exceptions that are thrown in the preceding try blocks. A catch block should not catch java.lang.Exception or java.lang.Throwable; this would lead to masking Errors or RuntimeExceptions. The only exception to this rule is if the module is some form of server that must continue operation irrespective of any fault.

Each catch block must either handle the Exception or pass it up by rethrowing it wrapped as another Exception. Catch blocks should always log the Exception to aid debugging.

Methods should never declare that they throw java.lang.Exception as this forces code that uses it to catch java.lang.Exception. Methods should always declare that they throw a specific sub class of Exception. In most cases this would be a custom Exception for the module.

try {

// Do something

}

catch (IOException e) {

// do something specific for this type of exception

e.printStackTrace();

throw new SpecificException(e);

}

finally {

// if anything always has to be done, even if an exception

// is caught, do it here...

}

Further, if the number of exceptions to be caught is large, or the processing involved is substantial, consider abstracting it out into a separate method.

The ‘finally’ part of a try/catch statement always called even if an exception is thrown in the ‘try’ part. If your code locks or grabs a resource, such as opening a file or a database connection, you should use a try/catch/finally construct to ensure the resource always freed. The general construct would be:

// grab resource here

try {

// use resource here

}

// repeat for all exceptions

catch (IOException e) {

// handle exception

}

finally {

// free the resource here

}

### Argument checking

A useful defensive programming technique is for all methods to check the validity of their arguments, throwing an IllegalArgumentException when an invalid argument is received. This is illustrated in the example below. Note that since IllegalArgumentException is derived from RuntimeException, it is not necessary to declare that the function may throw this exception.

public void addMoney(int total) {

if (total < 0) {

throw new IllegalArgumentException("'total' cannot be negative");

}

// ...do something with ‘total’

}

### Thread Safety

It is possible that Java code written on a project will be run in multi-threaded mode, the same instance of a class will be accessed by more than one client via a thread.. E.g. Servlets in WebLogic or JSF Action classes. For this reason thread safety must be a major consideration.

Consider the following class definition:

public class MyClass {

public void myFunction() {

int l = 10;

// Some code…

}

private static String NAME = null;

private float value = 1;

}

In this example, in a single JVM:

* all threads running through all instances of this class see the same value of the static variable NAME.
* all threads running though a particular instance of this class see the same value of the instance variable value.
* all threads get their own stack, so each thread has its own copy of any local variables such as ‘l’ in myFunction().

As described in section x.x, static variables should generally be avoided even in single-threaded situations, since they are shared across class instances. Further, in multi-threaded mode instance variables should be avoided unless you intend all threads running through a single class instance to see the same value. Put another way, two threads running through a single class instance cannot maintain different states, i.e. different values of static or instance class variables.

# Manage Applications

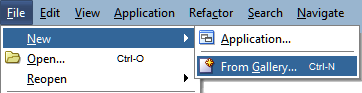
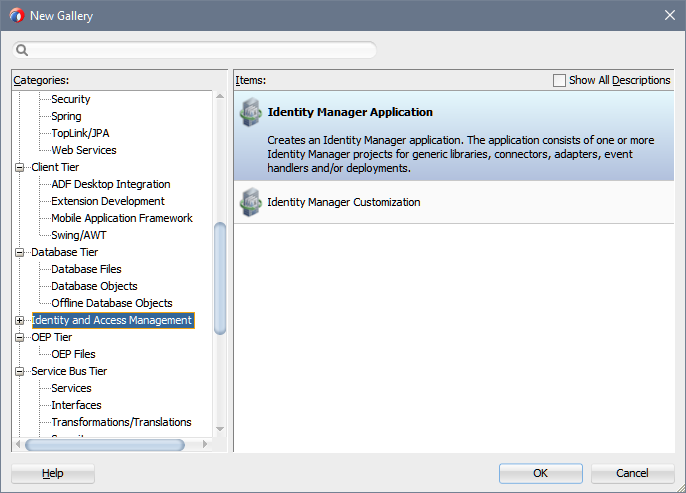
The activities for everything you need to create and work with the extension plug-in launched by navigate to **File** | **New...**.

No other opportunities to invoke a New Gallery dialog like **New...** from the **Application** menu or the **Application Navigator** exposed by the extension plug-in.

The reason for this restriction is that there is not any interface in Oracle JDeveloper that makes it possible to plug in our own Gallery Adapter. However, we need such an adapter to intercept the creation of applications and projects to configure those objects correctly with the minimal effort of manual tasks.

## Create an Application

To start the Create Identity Manager Application Wizard walkthrough:

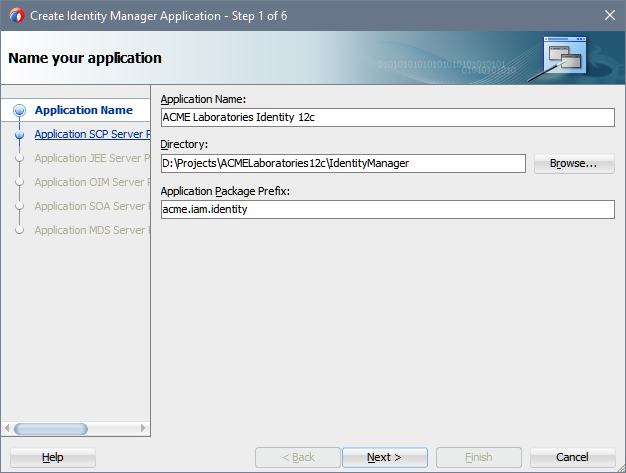
1. Open Oracle JDeveloper
2. Navigate to the menu **File** | **New** | **From Gallery...**This invokes the New Gallery dialogue to create objects of any sort, from applications or classes to JSPs and UML diagrams, and to set up and maintain your connections.
3. In the **Category** pane on the left scroll down to the **Oracle Identity and Access Management** and select this node. This displays the possible wizards you can start from here. Select the item **Identity Manager Application**.  
     
     
     
   This close the **New Gallery** dialog and starts the appropriate wizard that guides you through the required steps. The Wizard is performs several steps that are needed to create most of the configurations. After the wizard **Identity Manager Application** starts complete the steps below to create the application.

|  |  |
| --- | --- |
| note_edit | If a certain gallery item that belongs to a certain feature is not shown either the feature itself is disabled (see Configuring the Features) or the feature is not loaded or configured (see Configuring the Oracle Identity Manager Preferences).  Walkthrough the appropriate steps to change the configuration |

### Name your Application

On the step Name your Application you have to provide data for:

|  |  |
| --- | --- |
| **Application Name** | Enter a name for the application.  By default, an application name in the form ApplicationN is shown, where N is a number that increases sequentially from 1.  This is the filename that will be used for the application control file within the file system. The extension .jws is assumed, but not displayed.  Change this name to an appropriate name related to the customer environment. We will use  ACME Laboratories Identity 12c  for this purpose. |
| **Directory** | Enter a directory for the application or click **Browse...** to locate one.  Accordingly to the recommended File System Structure the directory name should be build by <Customer Name>/IdentityManager  Change this directory to  D:\Projects\ACMELaboratories12c\IdentityManager  for our purpose. |
| **Application Package Prefix** | It's required to enter a prefix to be used for packages created within this application.  The prefix will be applied to objects created in the initial project of an application.  Enter acme.iam.identity for our purpose. |



Click **Next**.

### Configure SCP Server Property settings

On the step Configure SCP Server Properties settings you have to provide data for:

* General Properties

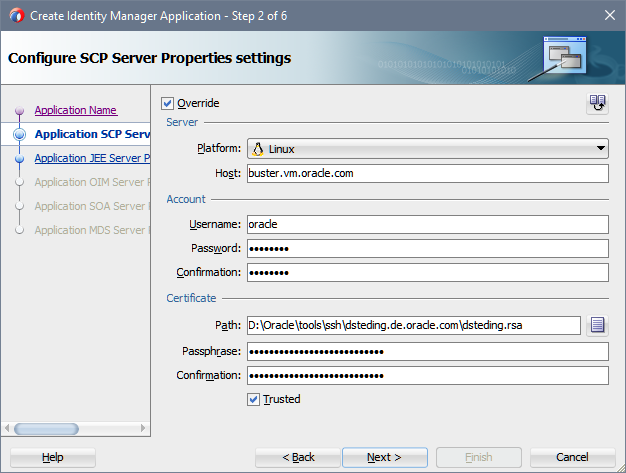
|  |  |
| --- | --- |
| **Platform** | The type of the target platform |
| **Host** |  |

* Account Properties

|  |  |
| --- | --- |
| **Username** |  |
| **Password** |  |
| **Confirmation** |  |

* Certificate Properties

|  |  |
| --- | --- |
| **Path** |  |
| **Passphrase** |  |
| **Confirmation** |  |
| **Trusted** |  |



Click **Next**.

### Configure JEE Server Property settings

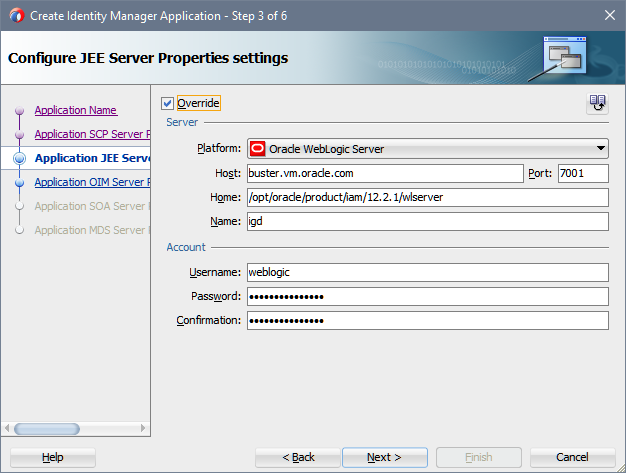
On the step Configure JEE Server Properties settings you have to provide data for:

* Server Properties

|  |  |
| --- | --- |
| **Platform** | The type of the target platform |
| **Host** |  |
| **Port** |  |
| **Home** |  |
| **Name** |  |

* Account Properties

|  |  |
| --- | --- |
| **Username** |  |
| **Password** |  |
| **Confirmation** |  |



Click **Next**.

### Configure OIM Server Property settings

On the step Configure OIM Server Properties settings you have to provide data for:

* Server Properties

|  |  |
| --- | --- |
| **Platform** | The type of the target platform |
| **Host** |  |
| **Port** |  |
| **Home** |  |
| **Name** |  |
| **Production** |  |

* Account Properties

|  |  |
| --- | --- |
| **Username** |  |
| **Password** |  |
| **Confirmation** |  |

* Authentication Configuration Properties

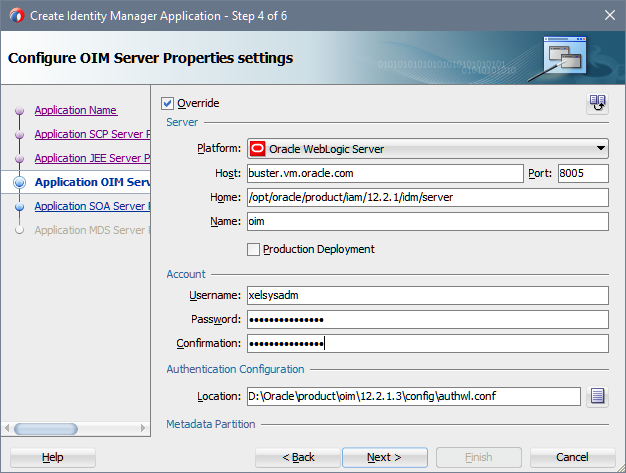
|  |  |
| --- | --- |
| **Location** |  |

* Metadata Partition Properties

|  |  |
| --- | --- |
| **Runtime** |  |
| **Sandbox** |  |

* MBean Properties

|  |  |
| --- | --- |
| **Metadata** |  |
| **Version** |  |



Click **Next**.

### Configure SOA Server Property settings

On the step Configure SOA Server Properties settings you have to provide data for:

* Server Properties

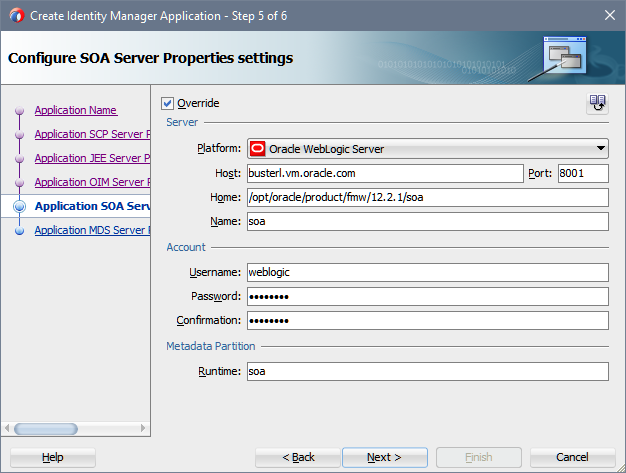
|  |  |
| --- | --- |
| **Platform** | The type of the target platform |
| **Host** |  |
| **Port** |  |
| **Home** |  |
| **Name** |  |

* Account Properties

|  |  |
| --- | --- |
| **Username** |  |
| **Password** |  |
| **Confirmation** |  |

* Metadata Partition Properties

|  |  |
| --- | --- |
| **Runtime** |  |



Click **Next**.

### Configure MDS Server Property settings

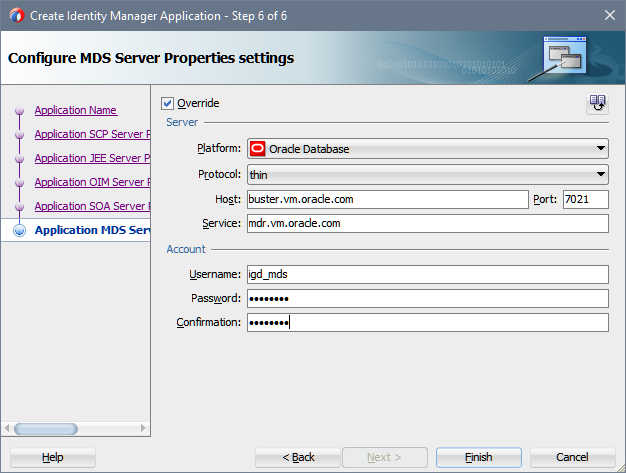
On the step Configure MDS Server Properties settings you have to provide data for:

* Server Properties

|  |  |
| --- | --- |
| **Platform** | The type of the target platform |
| **Host** |  |
| **Port** |  |
| **Home** |  |
| **Name** |  |

* Account Properties

|  |  |
| --- | --- |
| **Username** |  |
| **Password** |  |
| **Confirmation** |  |



Click **Next**.

## Review the Configuration

After the application created by the wizard you can review the configuration. You should find in your local file system following structure:

<Base Folder Workspaces>

+-- iam-12c-cpreferences.xml

|

+-- AMCELaboratories12c

| +-- wks-preferences.xml

| |

| +-- IdentityManager

| | +-- ACME Laboratories Identity 12c.jws

| | |

| | +-- jee-server.xml

| | +-- mds-server.xml

| | +-- oim-preferences.xml

| | +-- scp-server.xml

| | +-- oim-server.xml

# Manage Deployment Projects

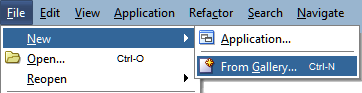
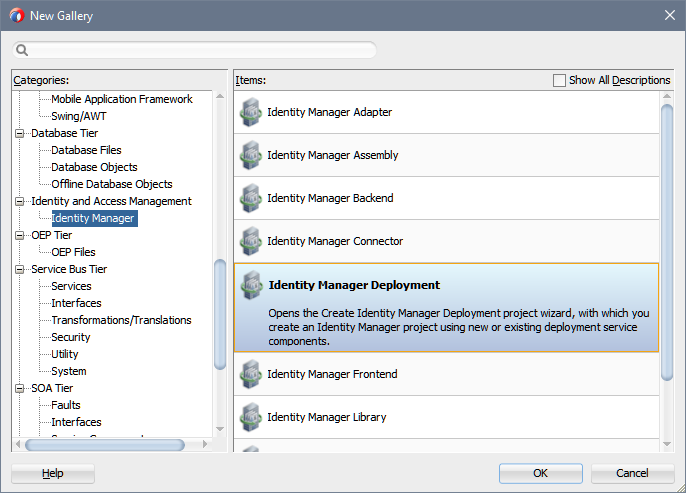
The activities for everything you need to create and work with the extension plug-in launched by navigate to **File** | **New...**.

No other opportunities to invoke a New Gallery dialog like **New...** from the **Application** menu or the **Application Navigator** exposed by the extension plug-in.

The reason for this restriction is that there is not any interface in Oracle JDeveloper that makes it possible to plug in our own Gallery Adapter. However, we need such an adapter to intercept the creation of applications and projects to configure those objects correctly with the minimal effort of manual tasks.

## Create a Deployment Project

To start the Create Identity Manager Deployment Project Wizard walkthrough:

1. Open Oracle JDeveloper
2. Navigate to the menu **File** | **New** | **From Gallery...**This invokes the New Gallery dialogue to create objects of any sort, from applications or classes to JSPs and UML diagrams, and to set up and maintain your connections.
3. It might happens that the New Gallery dialog opens with the tab **Current Project Technologies** selected.  
     
     
   In this case, you have to select the tab All Technologies at first.  
     
   In the **Category** pane on the left scroll down to the **Oracle Identity and Access Management** and expand this node. From this level select the option Identity Manager This displays the possible wizards you can start from here. Select the item **Identity Manager Application**.  
     
     
     
   This close the **New Gallery** dialog and starts the appropriate wizard that guides you through the required steps. The Wizard is performs several steps that are needed to create most of the configurations. After the wizard **Identity Manager Deployment** starts complete the steps below to create the project.

|  |  |
| --- | --- |
| note_edit | If a certain gallery item that belongs to a certain feature is not shown either the feature itself is disabled (see Configuring the Features) or the feature is not loaded or configured (see Configuring the Oracle Identity Manager Preferences).  Walkthrough the appropriate steps to change the configuration |

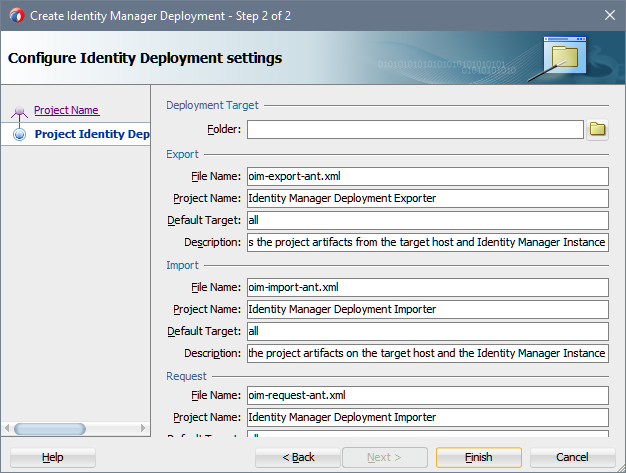
### Name your Project

On the step Name your Project you have to provide data for:

|  |  |
| --- | --- |
| **Project Name** | Enter a name for the project.  Per default, a project name in the form ProjectN is shown, where N is a number that increases sequentially from 1.  This is the filename that will be used for the project control file within the file system. The extension .jpr is assumed, but not displayed.  Change this name to an appropriate name related to the customer environment. We will use  Identity Manager 12c Deployment  for this purpose.  As you type the new name for the project to create you'll recognize that the field **Directory** is changed accordingly.    This we will revert to be aligned with the recommended File System Structure as explained below |
| **Directory** | Enter a directory for the application or click **Browse...** to locate one.It is easier to use the **Browse...** button to create the correct file system structure.  The dialog Choose Directory appears.    Per default the directory of the workspace you are creating the project within is already selected. Click on the **Create Directory** symbol in the upper right corner of the dialog to open the **Create New Directory** dialogue. Enter oimDeployment in the field **Directory Name**.    Click **OK** to close the popup.  Back on the dialog Choose Directory the newly created directory is selected.    Click **Select** to close the dialog and transfer the selection to the **Identity Manager Deployment** wizard. If you want to enter it manually type  <Base Folder Workspaces>\ACMELaboratories\IdentityManager\oimDeployment  in the field **Directory Name**.  At the end your screen should looks like    Click on **Next**. |

### Configure Identity Deployment settings

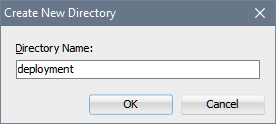
This page has already populated the default values for a single deployment project. To complete this page you have to specify the **Deployment Target** Location.



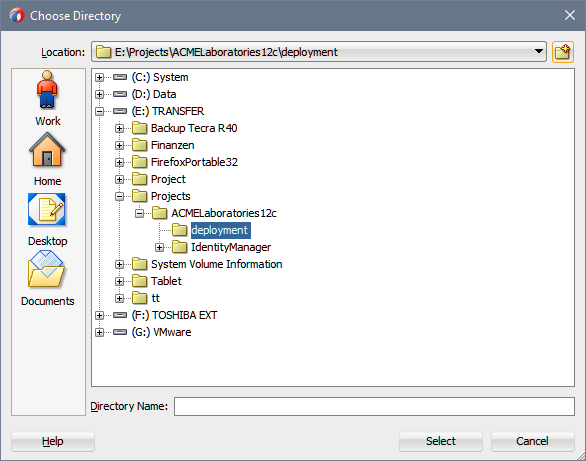
In Folder in the section **Deployment Target** you have to enter the location of your deployment in your local file system. The path can be either an existing on or you are creating a new one.

To create a new deployment path click on **Browse....**. The Create New Directory dialog appears again. Per default the directory of the workspace you created in Name your Application is selected. If not select the appropriate folder.

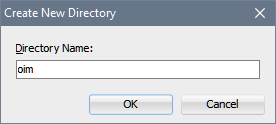
Click on the **Create Directory** symbol in the upper right corner of the dialog to open the Create New Directory dialog. Enter deployment in the field **Directory Name**.



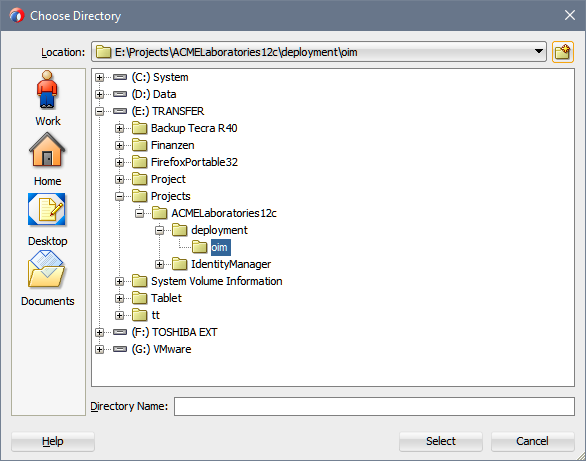
Click on **OK** to close the popup. Back on the dialog Choose Directory the newly created directory is selected.



Click again on the **Create Directory** symbol in the upper right corner of the dialog to open the Create New Directory dialog. Enter oim in the field **Directory Name**.

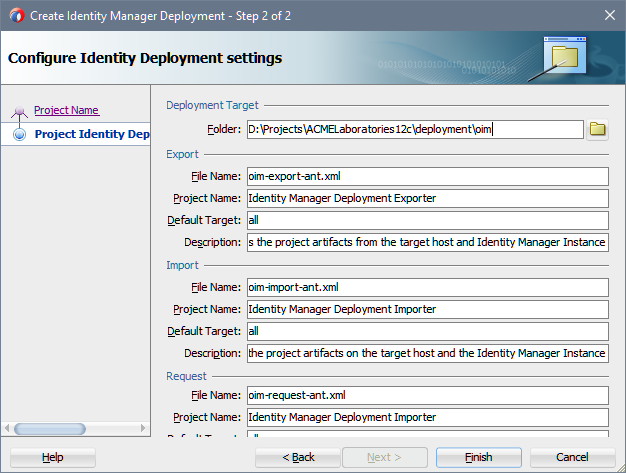


Click on **OK** to close the popup. Back on the dialog Choose Directory the newly created directory is selected.



Click **Select** to close the dialog and transfer the selection to the Identity Manager Deployment wizard.

After you completed the section Deployment Target review the further information provided on the page.



Click **Finish** to create the project and all artifacts necessary to use it.

## Review the Configuration

After the application was created by the wizard for you review the configuration.

File System Structure

You should find in your local file system following structure:

<Base Folder Workspaces>

+-- iam-12c-preferences.xml

|

+-- AMCELaboratories12c

| +-- wks-preferences.xml

| |

| +-- deployment

| | +-- oim

| |

| +-- IdentityManager

| | +-- ACME Laboratories Identity 12c.jws

| | |

| | +-- jee-server.xml

| | +-- mds-server.xml

| | +-- oim-context.xml

| | +-- oim-deployment.xml

| | +-- oim-preferences.xml

| | +-- oim-server.xml

| | +-- oim-targets.xml

| | +-- scp-server.xml

| | +-- soa-server.xml

| | |

| | +-- oimDeployment

| | | +-- Identity Manager 12c Deployment.jpr

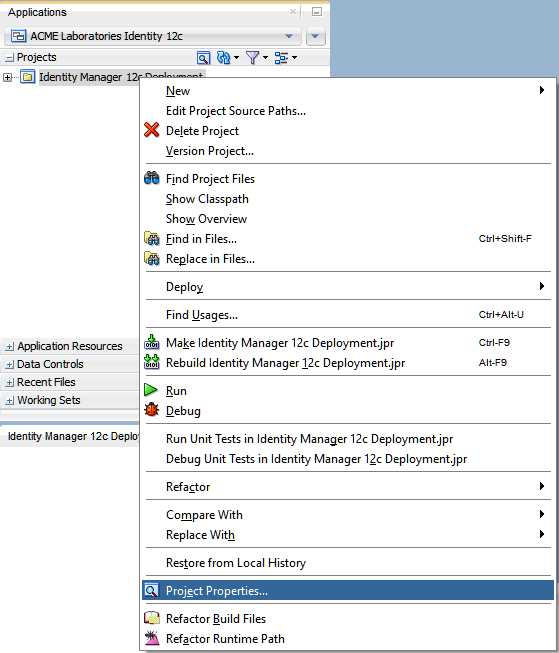
| | | +-- oim-export-ant.xml

| | | +-- oim-import-ant.xml

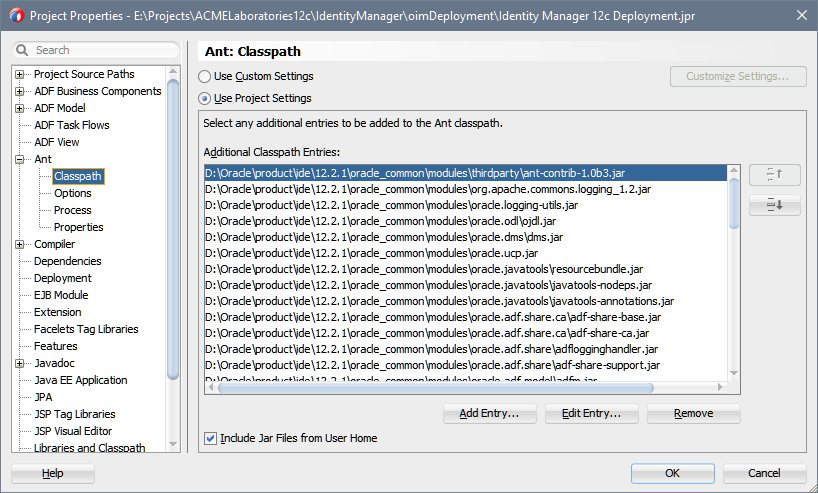
| | | +-- oim-request-ant.xml

Project Properties

The wizard has configured the Ant Classpath. You can review the changes by navigating to **Application** | **Project Properties…** in the main menu or in the context menu of the newly created project.



In the tree of the project properties select the node **Ant** and expand it to make the node **Classpath** visible. Select this node and you should see the configuration of the Ant classpath like the picture bellow.

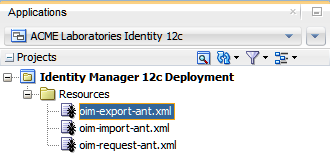


Click **OK** to close the Preferences dialog.

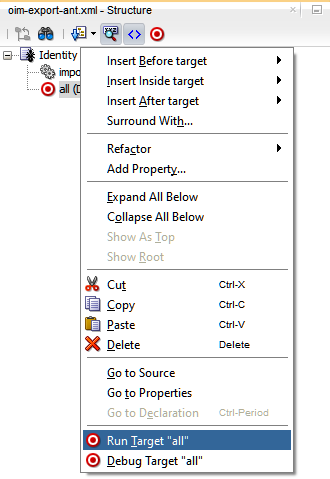
## Sanity Check

A quick sanity check should be performed on the created project.

Validate the Ant Build Script oim-export-ant.xml by selecting the document node on the node in the Application Navigator.

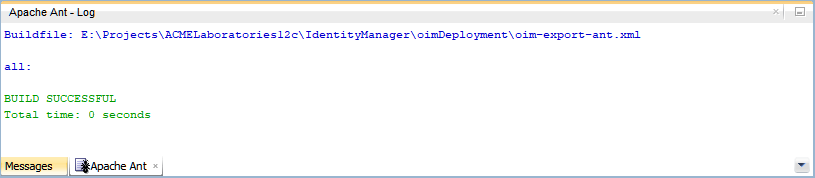


In the oim-export-ant.xml - Structure view below the Application Navigator make a right-mouse-click on the node all to open the context menu.



Click on the menu entry **Run Target "all"**.

In the Apache Ant - Log view you should recognize a successful completion of the task.



# Exporting Object from Oracle Identity Manager

You can export objects from your Oracle Identity Manager system and save them in an XML file. The Deployment Manager has an Export Wizard that lets you create your export file. Add objects by type, one type at a time, for example, roles, then forms, then processes, and so on.

Deployment Exporter enables you to export the objects that make up your Oracle Identity Manager configuration too. Like the Export Wizard, you use the Deployment Exporter to migrate a configuration from one deployment to another, for example, from a test to a production deployment, or to create a backup of your system.

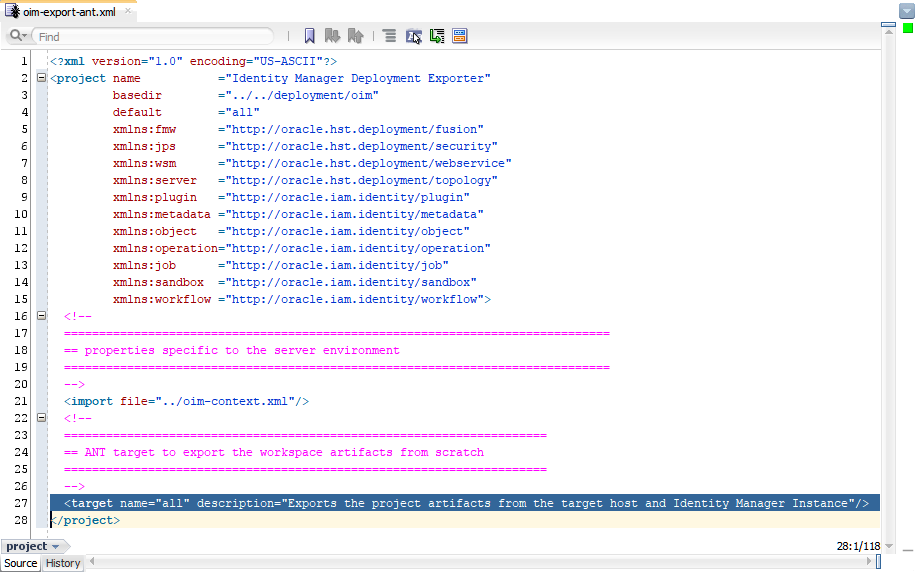
The Deployment Exporter saves your settings in an XML file. Use the Deployment Exporter to exchange Oracle Identity Manager Items between environments. These items include:

* Roles
* Organizations
* Access policies
* Attestation processes
* Authorization policies
* User metadata
* Roles and organization metadata
* Scheduled tasks
* Scheduled jobs
* IT resources
* Resource objects
* Lookup definitions
* Process forms
* Provisioning workflows and process task adapters
* Data object definitions
* Rules
* Notification templates
* Generic Technology Connector (GTC) providers
* Error codes
* System properties
* E-mail definitions
* Event Handlers
* Password policies
* Generic Technology Connectors
* IT resource definition
* Request templates
* Request datasets
* Approval policies

## About the Deployment Directory

## Configuring an Export Target

Open the file oim-export-ant.xml inside of Oracle JDeveloper by a mouse-double-click on the document node. This opens an Editor Window. The file will be empty.



Only the default target all is available.

### Create the Target to export System Objects

This export task will be export the definitions that you have created in “Lab 2 - Enterprise User lifecycle influenced by Delegated Administrators and End-users”. The objects to export are:

|  |  |
| --- | --- |
| **Lookup** | Lookup.Users.WorkExp |
| **User Metadata** | USR |
| **Organization** | ACME CAPITAL ACME HelpDesk ACME Public Finance ACME Mergers and Acquisitions ACME Taxation |
| **Role** | Role Owners Access Administrators ACME HelpDesk Administrators ACME Help Desk Administrators (remember the last role was pre seeded) |
| **Authorization Policy** | Role Owners - Manage Roles Access Administrators - Manage Role Membership Access Administrators - Search users HelpDesk CreateUser - Public Finance HelpDesk Create User - M & A HelpDesk SearchUser HelpDesk UpdateUser HelpDesk ModifyUser HelpDesk PasswordMgmt |
| **IT Resource** | System Configuration |

|  |  |
| --- | --- |
| note_edit | We recommend exporting each type of object to its own file. For example, all Lookup Definitions exported to acme-lookup-dm.xml.  At the time being, it might not be clear the reason to do so. But start thinking about fixes that has to be applied in a deployment that is running through UAT and that has to be transferred to a Production environment |

Create the target

In the open file oim-export-ant.xml create a target with the following attributes:

|  |  |
| --- | --- |
| **name** | 0101 |
| **description** | Exports the ACME Laboratories System Object Configuration |

It will be nice to your colleagues to put on top of the target a comment what the intended use of the target will be. The XML fragment should looks like:

<!--

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Purpose: Exports the Oracle Identity Manager System Object Configuration

\*\*

\*\* Synopsis: The tasks exports the changes made on System Object Configuration

\*\* like:

\*\* o System Configurations

\*\* o Lookup Definitions

\*\* o Roles

\*\* o IT Resources

\*\* o Organizations

\*\* o User Metadata

\*\* o Authorization Policies

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-->

<target name="0101" description="Exports the ACME Laboratories System Object Configuration">

</target>

### Declare the Export Task

The export task encapsulates activities related to the Deployment Manager Export capabilities of Oracle Identity Manager. The difference is how you have to remember what you have to export. Using the Deployment Manager you have to keep it in mind. We choose the approach to describe it in a file that than also performs the export for you.

An export task is declared as follow:

|  |  |
| --- | --- |
| **Namespace URI** | xmlns:object=http://oracle.iam.identity/object |
| **Task Name** | object:export |

The export task requires connection details and an export set that have to be exported. The connection details are already defined in the property files at the time you have created the application in “Create an Application”. The only thing what you have to is to use it.

In the open file oim-export-ant.xml create an export task with the following attributes:

|  |  |
| --- | --- |
| **contextRef** | oim-context |

The connection details specified by the attribute contextRef. This reference is specified in the configuration oim-context.xml. This file you have created in “Create a Deployment Project”.

To declare the export task you are extending the target definition 0101 by the xml fragment:

<object:export contextRef="oim-context">

</object:export>

### Declare the Export Set

The purpose of an Export Set is to accumulate object definitions of Oracle Identity Manager Objects that have to be exported to the same file. An Export Set is defined as a nested element to the export task. It cannot exists outside.

An export set is declared as follow:

|  |  |
| --- | --- |
| **Namespace URI** | xmlns:object=http://oracle.iam.identity/object |
| **Task Name** | object:exportSet |

In the open file oim-export-ant.xml create an export set with the following attributes:

|  |  |
| --- | --- |
| **exportFile** | 0101 systemObject/xml/acme-lookup-dm.xml |
| **description** | ACME Laboratories System Object Revision 1.0.0.0 |

To declare the export set you are extending the export task inside of the target definition 0101 by the xml fragment:

<object:exportSet exportFile = "0103 systemConfiguration/xml/acme-lookup-dm.xml"

description = "ACME Laboratories System Object Revision 1.0.0.0">

</object:exportSet>

### Checkpoint

Now you have a testable skeleton. You will not export anything but you can check if the configuration is valid overall.

Validate the Ant Build Script oim-export-ant.xml by selecting the document node on the node in the Application Navigator.

In the oim-export-ant.xml - Structure view below the Application Navigator make a right-mouse-click on the node all to open the context menu and click on the menu entry **Run Target “0101”**.

In the Apache Ant - Log view you should recognize a successful completion of the task.

# Appendix A

<Base Folder Workspaces>

+-- iam-preferences.xml

|

+-- CustomerWorkspaceFolder

| +-- wks-preferences.xml

| |

| +-- deployment

| | +-- oim

| | +-- 0100 systemFramework

| | +-- 0102 systemIntegration

| | |

| | +-- 0200 connectorBase

| | +-- 0201 connectorTarget XXX

| | +-- 0202 connectorTarget YYY

| | |

| | +-- 0300 connectorBase

| | +-- 0301 connectorTarget XXX

| | +-- 0302 connectorTarget YYY

| |

| +-- IdentityManager

| | +-- <Customer Application>.jws

| | |

| | +-- jee-server.xml

| | +-- mds-server.xml

| | +-- oim-context.xml

| | +-- oim-deployment.xml

| | +-- oim-preferences.xml

| | +-- oim-server.xml

| | +-- oim-targets.xml

| | +-- scp-server.xml

| | +-- soa-context.xml

| | +-- soa-deployment.xml

| | +-- soa-preferences.xml

| | +-- soa-server.xml

| | +-- soa-targets.xml

| | |

| | +-- <Deployment Project>

| | | +-- <Customer Deployment>.jpr

| | | |

| | | +-- ant

| | | +-- oim-export-ant.xml

| | | +-- oim-import-ant.xml

| | |

| | +-- <Adapter Project>

| | | +-- <Customer Adpater>.jpr

| | | |

| | | +-- ant

| | | | +-- oim-adapter-ant.xml

| | | | +-- oim-export-ant.xml

| | | | +-- oim-import-ant.xml

| | | |

| | | +-- bin

| | | +-- lib

| | | +-- src

| | |

| | +-- <Utility Project>

| | | +-- <Customer Utility>.jpr

| | | |

| | | +-- ant

| | | | +-- oim-adapter-ant.xml

| | | | +-- oim-export-ant.xml

| | | | +-- oim-import-ant.xml

| | | |

| | | +-- bin

| | | +-- lib

| | | +-- src

|

+-- OracleConsultingServices12c

| +-- wks-preferences.xml

| |

| +-- FoundationFramework

| | |

| | +-- hst-preferences.xml

| | +-- hst-targets.xml

| | |

| | +-- oim-server.xml

| | +-- scp-server.xml

| | |

| | +-- hstFoundation

| | |

| | +-- hst-foundation.version

| | |

| | +-- ant

| | | +-- hst-library-ant.xml

| | |

| | +-- bin

| | +-- lib

| | +-- src

| |

| +-- IdentityManager

| | |

| | +-- oim-preferences.xml

| | +-- oim-targets.xml

| | |

| | +-- jee-server.xml

| | +-- mds-server.xml

| | +-- oim-context.xml

| | +-- oim-server.xml

| | +-- scp-server.xml

| | +-- soa-server.xml

| | |

| | +-- oimAdapter

| | | |

| | | +-- ocs-adapter.version

| | | |

| | | +-- ant

| | | | +-- oim-adapter-ant.xml

| | | | +-- oim-export-ant.xml

| | | | +-- oim-import-ant.xml

| | | |

| | | +-- bin

| | | +-- lib

| | | +-- src

| | | +-- xml

| | |

| | +-- oimFoundation

| | | |

| | | +-- ocs-foundation.version

| | | |

| | | +-- ant

| | | | +-- oim-adapter-ant.xml

| | | | +-- oim-export-ant.xml

| | | | +-- oim-import-ant.xml

| | | |

| | | +-- bin

| | | +-- lib

| | | +-- src

| | | +-- xml

| | |

| | +-- oimUtility

| | | |

| | | +-- ocs-utility.version

| | | |

| | | +-- ant

| | | | +-- oim-adapter-ant.xml

| | | | +-- oim-export-ant.xml

| | | | +-- oim-import-ant.xml

| | | |

| | | +-- bin

| | | +-- lib

| | | +-- src

| | | +-- xml

# Glossary

| Term | Type | Definition or Synonym |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# Open and Closed Issues

## Open Issues

| ID | Issue | Resolution | Responsibility | Target Date | Impact Date |
| --- | --- | --- | --- | --- | --- |
| 001 | Startup of extension complains about unassigned technologies | No workaraound available |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Closed Issues

| ID | Issue | Resolution | Responsibility | Target Date | Impact Date |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |